# Learning to Optimize Combinatorial Functions

**Nir Rosenfeld** [1] **Eric Balkanski** [1] **Amir Globerson** [2] **Yaron Singer** [1]

## Abstract

Submodular functions have become a ubiquitous tool in machine learning. They are learnable from data, and can be optimized efficiently and with guarantees. Nonetheless, recent negative results show that optimizing learned surrogates of submodular functions can result in arbitrarily bad approximations of the true optimum. Our goal in this paper is to highlight the source of this hardness, and propose an alternative criterion for optimizing general combinatorial functions from sampled data. We prove a tight equivalence showing that a class of functions is optimizable if and only if it can be learned. We provide efficient and scalable optimization algorithms for several function classes of interest, and demonstrate their utility on the task of optimally choosing trending social media items.

## 1. Introduction

Submodular optimization is fast becoming a primary tool in machine learning. The power of submodularity as a model has been demonstrated in numerous applications, including document summarization (Lin & Bilmes, 2011), clustering (Gomes & Krause, 2010), active learning (Golovin & Krause, 2011; Guillory & Bilmes, 2011; Hoi et al., 2006), graph and network inference (Gomez Rodriguez et al., 2010; Rodriguez & Schölkopf, 2012; Defazio & Caetano, 2012), and information diffusion in networks (Kempe et al., 2003). Crucial to the success of these methods is the fact that optimizing submodular functions can be done efficiently and with provable guarantees (Krause & Golovin, 2014).

In many cases, however, the true function cannot be accessed, and instead a surrogate function is learned from data (Balkanski et al., 2017). To this end, *PMAC learning* (Balcan & Harvey, 2011) offers a framework for analyzing the learnability of submodular functions, as well as algorithms for learning in practice. Encouraging results show that in many cases submodular functions can be efficiently learned from data (Balcan & Harvey, 2011; Iyer et al., 2013; Feldman & Kothari, 2014; Feldman & Vondrak, 2016). A natural approach in this setting is to first learn a surrogate function from samples, and then optimize it, hoping that the estimated optimum will be close to the true one. A recent line of work has been devoted to this setting of *optimization from samples* (OPS) (Balkanski et al., 2016; 2017).

The main result of OPS is unfortunately discouraging: for maximizing a submodular function under a cardinality constraint, no algorithm can obtain a constant factor approximation guarantee given polynomially-many samples from *any* distribution (Balkanski et al., 2017). Thus, optimizing over learned surrogates does not provide any meaningful guarantees with respect to the true function.

The hardness of OPS is, however, a worst-case result. The hardness stems from the discrepancy between how the algorithm gains access to information (via samples) and how it is evaluated (globally). In contrast, machine learning objectives are typically concerned with *expected* outcomes, and are evaluated over the same distribution from which data is acquired (Valiant, 1984). In this paper, we build on this motivation and propose an alternative framework for optimizing from samples. The objective we propose, called *distributional optimization from samples* (DOPS), circumvents the above difficulties by considering a distribution-dependent objective. In general, a function class $\mathcal{F}$ is in $\alpha$-DOPS if an $\alpha$-approximation of the empirical argmax can be found with arbitrarily high probability using polynomially many samples, for any distribution $\mathcal{D}$ and for any $f \in \mathcal{F}$. Formally:

**Definition 1** ($\alpha$-DOPS)**.** *Let $\mathcal{F} = \{f : 2^{[n]} \to \mathbb{R}^+\}$ be a class of set functions over $n$ elements. We say that $\mathcal{F}$ is **$\alpha$-distributionally optimizable from samples** if there is an algorithm $\mathcal{A}$ that, for every distribution $\mathcal{D}$ over $2^{[n]}$, every $f \in \mathcal{F}$, and every $\epsilon, \delta \in [0, 1]$, when $\mathcal{A}$ is given as input a sample set $\mathcal{S} = \{(S^i, f(S^i))\}_{i=1}^M$ where $S^i \overset{iid}{\sim} \mathcal{D}$, with probability of at least $1 - \delta$ over $\mathcal{S}$ it holds that:*

$$\mathbb{P}_{\mathcal{T} \sim \mathcal{D}^m}\left[f\big(\mathcal{A}(\mathcal{T})\big) \geq \alpha \max_{S \in \mathcal{T}} f(S)\right] \geq 1 - \epsilon \qquad (1)$$

*where $\mathcal{T} = \{(S^j)\}_{j=1}^m$, $\mathcal{A}(\mathcal{T}) \in \mathcal{T}$ is the output of the algorithm, and $\mathcal{S}$ is of size $M \in poly(n, m, 1/\epsilon, 1/\delta)$.*

[1]Harvard University [2]Tel Aviv University. Correspondence to: Nir Rosenfeld <nirr@g.harvard.edu>.

*Proceedings of the $35^{th}$ International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. 

The criterion in Eq. (1) relaxes the OPS objective to hold in expectation over $\mathcal{D}$. This is achieved by replacing the entire combinatorial domain with a sampled subset $\mathcal{T}$ of size $m$, allowing for a distribution-agnostic notion of approximation. As $m$ increases, satisfying Eq. (1) is expected to be harder. When $m \to \infty$, DOPS recovers OPS.

Our first goal in this paper is to establish the hardness of DOPS. In general, classic approximation results do not necessarily transfer to statistical settings (Balkanski et al., 2017). Nonetheless, our main theoretical result establishes a tight equivalence between DOPS and PMAC learning (Balcan & Harvey, 2011), meaning that any $\mathcal{F}$ that is learnable is also optimizable, and vice versa. This demonstrates an intriguing link between learning and optimizing submodular functions, which are known to be PMAC-learnable (Balcan & Harvey, 2011). The equivalence result is constructive, and gives a general optimization algorithm which can utilize any PMAC learner as a black box for DOPS, and vice versa. While our main focus in this paper is on submodular functions, these results hold for any family of combinatorial functions.

In practice, however, optimizing via PMAC algorithms has several drawbacks (Balcan & Harvey, 2011; Feldman & Kothari, 2014; Feldman & Vondrak, 2016). Our second goal in this paper is hence to design an efficient and scalable DOPS algorithm for several classes of interest. Our algorithm optimizes a loss function whose minimization provides a sufficient condition for DOPS. We prove that the minimizer of the empirical loss can be used for recovering an approximate argmax. In this sense, the framework we propose is one in which the algorithm "learns to optimize". We show how the loss can be minimized efficiently and with guarantees for several submodular function classes, including coverage functions, cut functions, and unit demand.

An additional benefit of our approach is that it provides guarantees even when the output of the algorithm is restricted to a set of sampled alternatives. This setting is especially prevalent in cases where both sets and their values are generated by human users. For example, in the problem of influence maximization (Kempe et al., 2003), the goal is to choose a "seed" set of users such that, when exposed to certain content, will maximize its expected propagation. However, targeting arbitrary subsets of users is in most cases impossible, and the algorithm must choose between the sets of users sharing currently trending items. In the last part of the paper we demonstrate the empirical utility of our approach on this task using real data from Twitter.

## 2. Distributional optimization and learning

In this section we give a tight characterization of function classes in DOPS by showing that a class $\mathcal{F}$ is in DOPS if and only if it is PMAC-learnable. This involves two steps. In the first, we show that if $\mathcal{F}$ is $\alpha$-PMAC learnable with sample complexity $M_{\text{PMAC}}$, then it is $\alpha$-DOPS. We augment this result with tight sample complexity bounds for $\alpha$-DOPS. In the second part, we show that PMAC learnability is not only sufficient but also necessary for distributional optimization from samples. We show that if $\mathcal{F}$ is not $\alpha$-PMAC learnable, then it is not $(\alpha-\epsilon)$-DOPS for any constant $\epsilon > 0$, which is tight. This result is obtained by constructing a novel PMAC algorithm based on a DOPS black-box, and may thus be of separate interest in PMAC analysis. Overall, our results determine the hardness of DOPS by establishing a connection between the approximability and learnability of function classes.

We begin by reviewing the notion of PMAC learnability:

**Definition 2** (PMAC, Balcan & Harvey (2011))**.** *A class $\mathcal{F}$ is **$\alpha$-PMAC-learnable** if there is an algorithm such that for every distribution $\mathcal{D}$, every $f \in \mathcal{F}$, and every $\epsilon, \delta \in [0,1]$,*

$$\mathbb{P}_{S\sim\mathcal{D}}\left[\tilde{f}(S) \leq f(S) \leq \alpha\tilde{f}(S)\right] \geq 1-\epsilon \qquad (2)$$

*where the input of the algorithm is a set $\mathcal{S}$ of size $M \in poly(n, 1/\epsilon, 1/\delta)$, the output is a mapping $\tilde{f} : 2^{[n]} \to \mathbb{R}^+$, and Eq. (2) holds w.p. at least $1-\delta$ over $\mathcal{S}$.*

Intuitively, PMAC generalizes the standard notion of PAC learning by considering a loss which penalizes predictions that are not within a factor of $\alpha$ of their true value.

We are now ready to prove our main theoretical results.

### 2.1. If $\mathcal{F}$ is PMAC-learnable then $\mathcal{F}$ is in DOPS

We show that if $\mathcal{F}$ is $\alpha$-PMAC learnable with sample complexity $M_{\text{PMAC}}(n,\delta,\epsilon,\alpha)$, then it is $\alpha$-DOPS with sample complexity $M_{\text{PMAC}}(n,\delta,1-(1-\epsilon)^{1/m},\alpha)$, and this sample complexity is tight. A PMAC algorithm learns a surrogate function $\tilde{f}$. In our reduction, the corresponding DOPS algorithm simply outputs $\text{argmax}_{S\in\mathcal{T}}\tilde{f}(S)$. The technical part of this result is in showing the sample complexity tightness. Intuitively, the sample complexity is exactly the number of samples that are needed so that, with high probability, $\tilde{f}$ obtains a good approximation on all $S \in \mathcal{T}$. We begin by showing that $M_{\text{PMAC}}(n,\delta,1-(1-\epsilon)^{1/m},\alpha)$ is sufficient, which follows from the definition of PMAC.

**Theorem 1.** *Assume $\mathcal{F}$ is $\alpha$-PMAC-learnable with sample complexity $M_{PMAC}(n,\delta,\epsilon,\alpha)$, then $\mathcal{F}$ is $\alpha$-DOPS with sample complexity at most $M_{PMAC}(n,\delta,1-(1-\epsilon)^{1/m},\alpha)$, i.e.,*

$$M_{DOPS}(n,m,\delta,\epsilon,\alpha) \leq M_{PMAC}(n,\delta,1-(1-\epsilon)^{1/m},\alpha).$$

*Proof.* Let $f \in \mathcal{F}$, $\mathcal{D}$ be some distribution, $\mathcal{S} = \{(S_i, f(S_i))\}_{i=1}^M$ and $\mathcal{T} = \{S_i\}_{i=1}^m$ be the train and test sets, and $\mathcal{A}$ be an algorithm that constructs $\tilde{f}$ which $\alpha$-PMAC learns $f$ with sample complexity $M_{\text{PMAC}}(n,\delta,\epsilon,\alpha)$.

The DOPS algorithm that we analyze constructs $\tilde{f}$ with algorithm $\mathcal{A}$ using $\mathcal{S}$ and returns

$$\tilde{S}^\star = \operatorname*{argmax}_{S \in \mathcal{T}} \tilde{f}(S).$$

Fix $\epsilon, \delta > 0$ and $\alpha > 1$ and consider $M = M_{\texttt{PMAC}}(n, \delta, 1 - (1-\epsilon)^{1/m}, \alpha)$. By the definition of $\alpha$-PMAC, we get that with probability $1 - \delta$ over $\mathcal{S}$,

$$\Pr_{S \sim D}\left[\tilde{f}(S) \le f(S) \le \alpha \cdot \tilde{f}(S)\right] \ge (1-\epsilon)^{1/m}.$$

Next, we obtain

$$\Pr_{\mathcal{T}}\left[\tilde{f}(S) \le f(S) \le \alpha \cdot \tilde{f}(S) : \forall S \in \mathcal{T}\right] = \left(\Pr_{S \sim D}\left[\tilde{f}(S) \le f(S) \le \alpha \cdot \tilde{f}(S)\right]\right)^m \ge 1 - \epsilon.$$

where the equality is due to the sets $S \in \mathcal{T}$ being drawn i.i.d. from $\mathcal{D}$, and the inequality holds with probability $1-\delta$ over $\mathcal{S}$. We define $S^\star = \operatorname{argmax}_{S \in \mathcal{T}} f(S)$ and obtain that with probability $1 - \epsilon$ over $\mathcal{T}$ and $1 - \delta$ over $\mathcal{S}$,

$$f(\tilde{S}^\star) \ge \tilde{f}(\tilde{S}^\star) \ge \tilde{f}(S^\star) \ge \alpha^{-1} f(S^\star).$$

We conclude that with $M = M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha)$,

$$f(\tilde{S}^\star) \ge \frac{1}{\alpha} \cdot \max_{S \in \mathcal{T}} f(S)$$

with probability $1 - \epsilon$ over $\mathcal{T}$ and $1 - \delta$ over $\mathcal{S}$. □

For tightness, we give an information-theoretic lower bound by constructing a difficult class $\mathcal{F}$ that cannot be in $\alpha$-DOPS with less than $M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha)$ samples.

**Theorem 2.** *For all $\alpha > 1$ and $\epsilon, \delta > 0$, for $m$ sufficiently large, there exists a family of functions $\mathcal{F}$ and a function $M_{\texttt{PMAC}}(\cdot)$ such that*

- *for all $\epsilon', \delta' > 0$: $\mathcal{F}$ is $\alpha$-PMAC-learnable with sample complexity $M_{\texttt{PMAC}}(n, \delta', \epsilon', \alpha)$, and*
- *given strictly less than $M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha)$ samples, $\mathcal{F}$ is not $\alpha$-DOPS, i.e.,*

$$M_{\texttt{DOPS}}(n, m, \delta, \epsilon, \alpha) \ge M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha).$$

*Proof Sketch (see supp. material for full proof).* For each $f$ in the difficult $\mathcal{F}$, only a single set $S^\star$ has a high value, while all others have low values. We consider a uniformly random function $f \in \mathcal{F}$ and the corresponding randomized subclass $\mathcal{F}' \subseteq \mathcal{F}$ which consists of all functions $f'$ such that $S^\star$ is in the test set but not in the train set.

Informally, an algorithm which aims to optimize $f \in \mathcal{F}'$ cannot use the train set to learn which $S \in \mathcal{T}$ is $S^\star$. More precisely, if $f \in \mathcal{F}'$, the decisions of the algorithm are *independent* of the randomization of $f$, conditioned on $f \in \mathcal{F}'$. Thus, if $f \in \mathcal{F}'$, the algorithm does not obtain an $\alpha$-approximation because of the gap between the value of $S^\star$ and the other sets.

We construct $\mathcal{F}$ and $\mathcal{D}$ such that $S^\star$ is in the test set w.p. greater than $1 - \epsilon$. This implies that to satisfy DOPS, the algorithm must observe enough samples so that $S^\star$ is in the train set w.p. at least $1 - \delta$. We then argue that this number of samples is at least $M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha)$. □

### 2.2. If $\mathcal{F}$ is not PMAC-learnable then $\mathcal{F}$ is not in DOPS

A simple intuition for Theorem 1 is that if one can accurately predict the values of all $S \in \mathcal{T}$, then it is possible to find the empirical argmax. The main result in this section, which is perhaps less intuitive, shows that the reverse implication also holds. Namely, if one can find the the empirical argmax, then it is possible to infer the values of *all* sets in $\mathcal{T}$. The contrapositive of this result is that if $\mathcal{F}$ is not PMAC-learnable, then $\mathcal{F}$ is not in DOPS. Combining both results provides a full characterization of distributional optimization in terms of learnability.

To construct a PMAC learner from a DOPS algorithm, we first randomly partition $\mathcal{S}$ into "train" and "test" sets. We then train the DOPS algorithm on the train set, and use it to generate pairwise comparisons with test elements. The learned value for $S$ is given by the maximum value of a test sample that $S$ "beats" (via the inferred comparisons). At a high level, the analysis uses the DOPS guarantees and a bucketing argument to satisfy the PMAC requirements.

**Theorem 3.** *Let $\mu = \max_S f(S) / \min_{S: f(S) > 0} f(S)$, $c$ be any constant such that $1 \le \alpha \le c$, and $M_\mu = \frac{8 \log \mu}{\epsilon \log c}\left(\frac{1}{\epsilon} + 2\log\left(\frac{1}{\delta}\right)\right)$. If a class $\mathcal{F}$ is in $\alpha$-DOPS with sample complexity $M_{\texttt{DOPS}}(n, m, \epsilon, \delta, \alpha)$, then it is $\alpha$-PMAC-learnable with sample complexity $M_\mu + M_{\texttt{DOPS}}(n, 2, \epsilon/M_\mu, \delta/M_\mu, \alpha/c)$, i.e.,*

$$M_{\texttt{PMAC}}(n, \epsilon, \delta, \alpha) \ge M_\mu + M_{\texttt{DOPS}}(n, 2, \epsilon/M_\mu, \delta/M_\mu, \alpha/c).$$

*Proof.* Fix $\epsilon, \delta > 0$ and $\alpha > 1$. Let $\mathcal{S} = \{(S_i, f(S_i))\}_{i=1}^M$ be the samples from $\mathcal{D}$ that are given as input. We partition the samples in $\mathcal{S}$ uniformly at random into $\mathcal{S}_1$ and $\mathcal{S}_2$ of sizes $M_1$ and $M_2$, respectively. For some $S \sim \mathcal{D}$, the goal is to predict $\tilde{f}(S)$ such that $\tilde{f}(S) \le f(S) \le \alpha \cdot \tilde{f}(S)$.

For each $S_i \in \mathcal{S}_2$, define $\mathcal{S}_{2,i} := \{S_i, S\}$. Since $\mathcal{F}$ is in DOPS, with $M_1 = M_{\texttt{DOPS}}(n, 2, \epsilon/M_2, \delta/M_2, \alpha/c)$ samples, the algorithm outputs $S_i^\star \in \mathcal{S}_{2,i}$ such that with probabilities $1 - \delta/M_2$ over $\mathcal{S}_1$ and $1 - \epsilon/M_2$ over $\mathcal{S}_{2,i}$,

$$f(S_i^\star) \ge \frac{\alpha}{c} \max(f(S), f(S_i)).$$

By a union bound, this holds for all $i \in M_2$ with probability $1 - \delta$ over $\mathcal{S}_1$ and probability $1 - \epsilon$ over $S$ and $\mathcal{S}_2$.

We say that $S$ "beats" $S_i$ if the $\alpha$-DOPS algorithm outputs $S$ when given $\mathcal{S}_{2,i}$. Let $\mathcal{S}_2^-$ be the collection of sets $S_i$ in $\mathcal{S}_2$ such that $S$ beats $S_i$. The learning algorithm is

$$\tilde{f}(S) = \frac{c}{\alpha} \cdot \max_{S_i \in \mathcal{S}_2^-} f(S_i).$$

Let $f_{\min} = \min_S f(S)$ and $f_{\max} = \max_S f(S)$. We partition the sets into buckets defined as follows:

$$B_i := \{S : f_{\min} \cdot c^{i-1} \leq f(S) < f_{\min} c^i\}$$

for $i \geq 1$ and $B_0 = \{S : f(S) = 0\}$. With $\beta := \log \mu / \log c$ buckets, all sets $S$ are in a bucket since $f_{\min} \leq f(S) \leq f_{\max}$. We define a bucket $B_i$ to be dense if a random set $S \sim D$ has non-negligible probability to be in $B_i$, otherwise it is sparse. More precisely, $B_i$ is dense if $\Pr_{S \sim D}[S \in B_i] \geq \epsilon / 2\beta$.

The set $S$ is in a dense bucket $B_i$ with probability at least $1 - \frac{\epsilon}{2}$ since there are at most $\beta$ buckets that are not dense and $S$ is in each of them with probability at most $\frac{\epsilon}{2\beta}$ by the definition of dense bucket. With $m$ samples, the expected number of samples in $B_i$ is at least $m\frac{\epsilon}{2\beta}$ and by a standard concentration bound,

$$\Pr\left[|B_i| \leq \frac{m}{2}\frac{\epsilon}{2\beta}\right] \leq e^{-\frac{m\epsilon}{16\beta}}$$

We assume that $|B_i| \geq \frac{m}{2}\frac{\epsilon}{2\beta}$ for the remainder of the proof. There is at most one set in bucket $B_i$ that is beaten by all the other sets. Since the set $S$ has equal probability to be any of the sets in $B_i$,[1] there is at least one other set $S^-$ in $B_i$ which $S$ beats with probability $1/|B_i| \leq 4\beta/m\epsilon$.

With $\delta \geq e^{-\frac{m\epsilon}{16\beta}}$ (and hence $m \geq \frac{\log(1/\delta)16\beta}{\epsilon}$), with probability of at least $1-\delta$, the number of samples in $B_i$ is at least $m\frac{\epsilon}{4\beta}$. With $\epsilon/2 \geq 4\beta/m\epsilon$ (and hence $m \geq 8\beta/\epsilon^2$), with probability of at least $1-\epsilon$ over $S \sim \mathcal{D}$, $S$ is in a dense bucket and beats at least one other $S^- \in \mathcal{S}_2^-$ in that bucket.

We get that:

$$\tilde{f}(S) = \frac{c}{\alpha} \cdot \max_{S_i \in \mathcal{S}_2^-} f(S_i) \geq \frac{c}{\alpha} \cdot f(S^-) \geq \frac{1}{\alpha} \cdot f(S)$$

where the equality is by the definition of $\tilde{f}(S)$, the first inequality is since $S^- \in \mathcal{S}_2^-$, and the last is since $S$ and $S^-$ are in the same bucket. We also have

$$f(S) \geq \frac{\alpha}{c} \cdot \max_{S_i \in \mathcal{S}_2^-} f(S_i) = \tilde{f}(S)$$

where the inequality is by the definition of $\mathcal{S}_2^-$ and the equality by definition of $\tilde{f}(S)$. Thus, $\tilde{f}(S) \leq f(S) \leq \alpha \tilde{f}(S)$ and with $M_2 = m \geq \frac{8 \log \mu}{\epsilon \log c}\left(\frac{1}{\epsilon} + 2\log\left(\frac{1}{\delta}\right)\right) = M_\mu$, the sample complexity is $M_\mu + M_{\text{DOPS}}(n, 2, \epsilon/M_\mu, \delta/M_\mu, \alpha/c)$. □

[1] We assume that the DOPS algorithm breaks ties in a consistent manner, i.e., it cannot be adversarial and break ties depending on whether $S$ is the set we wish to learn or if $S \in \mathcal{S}_2$.

**Algorithm 1** DOPS$(\mathcal{S} = \{(S_i, z_i)\}_{i=1}^M,\ m, \alpha)$

1: Randomly partition $[M]$ into $N = \lfloor \frac{M}{m} \rfloor$ sets $A_1, \ldots, A_N$
2: Create $m$-tuple sample set $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{S}^i, \mathbf{z}^i)\}_{i=1}^N$ from $\mathcal{S}$ where $\boldsymbol{S}^i = \{S_j\}_{j \in A_i}$ and $\mathbf{z}^i = \{z_j\}_{j \in A_i}$
3: Compute $\alpha(\mathbf{z}^i) = \{y \in [m] : \mathbf{z}^i_y \geq \alpha \max \mathbf{z}^i\} \quad \forall\, i \in [N]$
4: $\hat{\theta} = \operatorname{argmin}_{\theta \in \Theta} \sum_{i=1}^N \max_y [\mathbb{1}_{\{y \notin \alpha(\mathbf{z}^i)\}} + f_\theta(\boldsymbol{S}^i_y) - \psi_\theta(\boldsymbol{S}^i, \mathbf{z}^i)]_+$ where $\psi_\theta(\boldsymbol{S}, \mathbf{z}) = \frac{1}{|\alpha(\mathbf{z})|} \sum_{y \in \alpha(\mathbf{z})} f_\theta(\boldsymbol{S}_y)$
5: Return $h_{\hat{\theta}}(\mathcal{T}) = \operatorname{argmax}_{S \in \mathcal{T}} f_{\hat{\theta}}(S)$

## 3. Learning to Optimize at Scale

In this section we give an efficient DOPS algorithm that applies to several interesting parametric submodular subclasses $\mathcal{F}_\Theta = \{f_\theta : \theta \in \Theta\}$. Our general technique includes two steps. First, we identify a loss function whose minimization provides a sufficient condition for DOPS (Eq. (1)), but is in general hard to optimize. Then, we show that for the function classes we consider, a transformation of the inputs reveals structure which can be exploited for efficiently optimizing a convex surrogate loss. Note that in principle, due to Thm. 1, any PMAC algorithm can be used for DOPS. This, however, has several practical disadvantages, which we comment on in Sec. 3.5.

We begin by illustrating our approach for coverage functions with parametric weights. We then describe our algorithm, prove its correctness, and show how it can be applied to other classes such as graph cuts, unit demand, and coverage functions with parametric cover sets.

### 3.1. Learning to optimize coverage functions

Coverage functions are a simple but important class of submodular functions, and have been used in applications such as computational linguistics (Sipos et al., 2012), algorithmic game theory (Dughmi & Vondrák, 2015), and influence maximization in social networks (Kempe et al., 2003). Let $U$ be a ground set of $d$ items, and $\mathcal{C} = \{C_1, \ldots, C_n\}$ a collection of subsets where $C_i \subseteq U$. For a set of non-negative item weights $\theta = \{\theta_1, \ldots, \theta_d\}$, a function $f_\theta : 2^{[n]} \to \mathbb{R}$ is a *coverage function* if:

$$f_\theta(S) = \sum_{u \in C(S)} \theta_u, \qquad C(S) = \bigcup_{i \in S} C_i \tag{3}$$

While apparently simple, coverage functions are quite expressive, and optimizing them from samples is known to be hard (Balkanski et al., 2017). One reason is that, as a

function of their inputs $S$, coverage functions can be highly non-linear. Meanwhile, as a function of their *parameters*, they become linear via a simple transformation of the inputs:

$$f_\theta(S) = \langle \phi(S), \theta \rangle, \qquad \phi_u(S) = \mathbb{1}_{\{\exists\, i \in S \text{ s.t. } u \in C_i\}} \tag{4}$$

This structure allows our algorithm to efficiently find the approximate empirical argmax of any given $\mathcal{T}$ with high probability. The output of the algorithm is a function $h \in \mathcal{H}$ for choosing one $S$ out of the $m$ candidates in $\mathcal{T}$, where:

$$\mathcal{H} = \{h_\theta(\mathcal{T}) = \operatorname*{argmax}_{S \in \mathcal{T}} f_\theta(S) : \theta \in \Theta\} \tag{5}$$

In this sense, our method "learns" how to optimize over collections of size $m$.

### 3.2. Algorithm

Pseudocode of our DOPS algorithm is given in Algorithm 1. The following theorem establishes its correctness:

**Theorem 4.** *Let $m \in \mathbb{N}$ and $\epsilon, \delta \in [0,1]$, and let $f = f_{\theta^*}$ with $\theta^* \in \Theta$. For a given $\alpha > 0$, let $h$ be the output of Algorithm 1 when given $\mathcal{S} = \{(S^i, z^i)\}_{i=1}^M$, $m$, and $\alpha$ as input, where $z = f_\theta(S)$ and $S \overset{iid}{\sim} \mathcal{D}$. Then, with probability of at least $1-\delta$ over $\mathcal{S}$, it holds that:*

$$\mathbb{P}_{\mathcal{T} \sim \mathcal{D}^m}\left[ f\big(h(\mathcal{T})\big) \geq \alpha \max_{S \in \mathcal{T}} f(S) \right] \geq 1 - \epsilon \tag{6}$$

*for $M \geq \tilde{O}(m(kB/\epsilon)^2)$ with $k = \max|S|$, $B = \|\theta^*\|_2$.*

The following proof holds for any class of functions that can be made linear in their parameters under some representation. This includes the coverage functions in Sec. 3.1 as well as the classes we consider in Sec. 3.3.

*Proof.* We begin with some notation. Let $\boldsymbol{S} = \{S_1, \ldots, S_m\}$ be a set of $m$ examples with corresponding values $\mathbf{z} = \{z_1, \ldots, z_m\}$ where $z_y = f(S_y)$. Algorithm 1 returns a function $h$ that chooses a set $S_y \in \boldsymbol{S}$. It will be convenient to instead view $h$ as a mapping from $\boldsymbol{S}$ to indices $y \in [m]$. Denote the set of $\alpha$-approximate solutions by:

$$\alpha(\mathbf{z}) = \{y \in [m] : z_y \geq \alpha \max \mathbf{z}\} \tag{7}$$

Our analysis makes use of the following loss function:

$$\Delta_\alpha(\mathbf{z}, y) = \mathbb{1}\{y \notin \alpha(\mathbf{z})\} \tag{8}$$

Eq. (8) is useful since $\mathcal{L}(h) := \mathbb{E}[\Delta_\alpha(\mathbf{z}, h(\boldsymbol{S}))] \leq \epsilon$ implies that $h$ satisfies Eq. (6). We therefore focus on bounding $\mathcal{L}(h)$. As we do not have access to $\mathcal{D}$, our algorithm chooses an $h \in \mathcal{H}$ which instead minimizes the empirical loss. Note that while $\Delta_\alpha$ is defined over $m$-tuples, $\mathcal{S}$ contains individual sets. To ensure a consistent empirical loss, we randomly partition $[M]$ into $N = M/m$ distinct sets $A_1, \ldots, A_N$, and define an $m$-tuple sample set $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{S}^i, \mathbf{z}^i)\}_{i=1}^N$, where $\boldsymbol{S}^i = \{S_y\}_{y \in A_i}$ and $\mathbf{z}^i = \{z_y\}_{y \in A_i}$. The loss is now:

$$\hat{\mathcal{L}}(h; \boldsymbol{\mathcal{S}}) = \frac{1}{N} \sum_{i=1}^N \Delta_\alpha(\mathbf{z}^i, \hat{y}^i), \qquad \hat{y}^i = h(\boldsymbol{S}^i) \tag{9}$$

Since $\Delta_\alpha$ is not convex, the algorithm instead optimizes a surrogate convex upper bound. There are many ways to do this; here we use an average hinge surrogate:

$$\max_{y \in [m]} \left[\Delta_\alpha(\mathbf{z}^i, y) + f_\theta(\boldsymbol{S}^i_y) - \psi_\theta(\boldsymbol{S}^i, \mathbf{z}^i)\right]_+ \tag{10}$$

where $[a]_+ = \max\{0, a\}$ and:

$$\psi_\theta(\boldsymbol{S}, \mathbf{z}) = \frac{1}{|\alpha(\mathbf{z})|} \sum\nolimits_{y \in \alpha(\mathbf{z})} f_\theta(\boldsymbol{S}_y) \tag{11}$$

Eq. (10) is similar in spirit to the loss in (Lapin et al., 2015), and is tight w.r.t. Eq. (9) whenever $\hat{\mathcal{L}} = 0$, Intuitively, minimizing Eq. (10) pushes $\theta$ towards values for which the true argmax is scored higher than all others by a margin. Note that the average in Eq. (11) can be replaced with a max to attain a tighter (though no longer convex) surrogate.

Since $\boldsymbol{\mathcal{S}}$ is labeled by some $f_{\theta^*} \in \mathcal{F}_\Theta$, we have that $\mathcal{L}(h_{\theta^*}) = 0$. This means that there is some $\theta \in \Theta$ such that with $\hat{\mathcal{L}}(h_\theta; \boldsymbol{\mathcal{S}}) = 0$, and due to the tightness of Eq. (10), $\tilde{\mathcal{L}}(h_\theta; \boldsymbol{\mathcal{S}}) = 0$ as well. This is sufficient for applying the following generalization bound (Collins, 2004):

$$\mathcal{L}(h) \leq O\left(\sqrt{\frac{m}{M}\left((kB\log M)^2 + \log\frac{1}{\delta}\right)}\right) \tag{12}$$

Plugging in $M$ gives $\mathcal{L}(h) \leq \epsilon$, concluding the proof. □

Eq. (10) is convex whenever $f_\theta$ is linear in $\theta$ for some representation $\phi$. This holds for coverage functions (Eq. (4)) as well as for the other classes we consider in Sec. 3.3. Eq. (10) can then be optimized using standard convex solvers, or with highly efficient and scalable solvers such as the cutting plane method of Joachims et al. (2009).

### 3.3. Other submodular classes

We now discuss how our method can be extended to other submodular function classes. For each class, we give a transformation $\phi$ of the inputs under which the function becomes linear in its parameters. Thm. 4 and Algorithm 1 can then be applied with the appropriate $f_\theta(S) = \langle \phi(S), \theta \rangle$.

**Graph $k$-cuts:** Let $G = (V, E)$ be an undirected graph, and let $\theta \in \mathbb{R}_+^{|E|}$ be edge weights. For a partition $P \in [k]^{|V|}$ of the nodes into $k$ groups, its value is given by:

$$f_\theta(P) = \frac{1}{2} \sum_{\substack{(u,v) \in E \\ P_u \neq P_v}} \theta_{uv}$$

While $k$-cut functions are known to be hard to optimize over $P$, they become linear in $\theta$ with the transformation:

$$\phi_{uv}(P) = \mathbb{1}\{P_u \neq P_v\} \qquad \forall\,(u,v) \in E$$

**Unit demand:** Let $\theta \in \mathbb{R}^n_+$ be a set of item weights. The value of a subset $S \subseteq [n]$ is given by:

$$f_\theta(S) = \max_{u \in S} \theta_u$$

Although it is possible to write $f_\theta = \langle \theta, \phi(S) \rangle$ with $\phi_u(S) = \mathbb{1}_{\{\theta_u \geq \theta_v\ \forall v \in S\}}$, this representation requires $\theta$, which is unknown. Nonetheless, a similar data-dependent construction can still be used to obtain some $\theta'$ which minimizes the loss. To see why, let $\bar{S} \in \mathcal{S}$ be the set with the highest value $f_\theta(\bar{S})$ in S. For this $\bar{S}$, there must exist some $u \in \bar{S}$ that is not in any other $S \in \mathcal{S}$ with $f_\theta(S) < f_\theta(\bar{S})$. By setting $\phi_v(\bar{S}) = \mathbb{1}_{\{u=v\}}$ and $\theta'_u = f_\theta(\bar{S})$, we ensure that $f_\theta(\bar{S}) = \langle \theta', \phi(\bar{S}) \rangle$. Note that this does not necessarily imply that $\theta'_u = \theta_u$. In a similar fashion, by setting:

$$\phi_u(S_i) = \mathbb{1}\{u \in S_i \wedge \nexists\, j \neq i \text{ s.t. } u \in S_j \wedge z_j < z_i\}$$

for every $i \in M$, we get that $f_\theta(S^i) = \langle \theta', \phi(S^i) \rangle$ for *some* $\theta'$, which guarantees $\hat{\mathcal{L}} = 0$. Note that generalization here concerns $\phi$ as applied to examples in both $\mathcal{S}$ and $\mathcal{T}$.

**Coverage with parametrized cover sets:** Let $U = [N]$ be a ground set of items with unit weights. The parameters are a collection item subsets $\{C_1, \ldots, C_n\}$ with $C_i \subseteq U$. We use $\xi_{iu} = \mathbb{1}\{u \in C_i\}$ and denote the maximal overlap by $d = \max_u \sum_i \xi_{iu}$. For a subset $S \in [n]$, its value is:

$$f_C(S) = \left| \bigcup_{i \in S} C_i \right|$$

While $f_C$ is not linear over $C$, it can be linearized over a different parameterization. For $x_i = \mathbb{1}\{i \in S\}$, we have:

$$f_C(S) = \sum_{u \in \Omega} \left(1 - \prod_{i=1}^{n} (1 - x_i \xi_{iu})\right)$$

Since $f_C$ is a polynomial of degree at most $d$, the explicit size of $\phi$ (and hence of the corresponding $\theta$) is $n^d$. For computational efficiency, we can consider the dual form and implicitly define $\phi$ via the kernalized inner product:

$$\langle \phi(S), \phi(S') \rangle = \left(\langle x_S, x_{S'} \rangle + 1\right)^d$$

### 3.4. Reducing the sample-complexity cost of $m$

Interestingly, at the cost of a small additional additive error, the dependence of the generalization bound on $m$ can be removed by considering an alternative loss function. Fix some $q \in [0,1]$. Given $\mathcal{S}$, define $Q$ to be the set of examples in the top $q$-quantile. The idea here is to learn $\theta$ so that $f_\theta$ will score top-quantile examples $S \in Q$ above low-quantile examples $S \notin Q$. The corresponding loss is therefore defined over example pairs:

$$\Delta_q(S, S', f_\theta) = \begin{cases} \mathbb{1}_{\{f_\theta(S) < f_\theta(S')\}} & \text{if } S \in Q \wedge S' \notin Q \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Note that, in a similar fashion to $\Delta_\alpha$, the empirical loss $\hat{\mathcal{L}}_q$ over $\Delta_q$ can be optimized efficiently, and the optimal $\theta$ gives $\hat{\mathcal{L}}_q = 0$. For any $\boldsymbol{S} \in \boldsymbol{\mathcal{S}}$, the probability of having at least one $S \in \boldsymbol{S} \cap Q$ is $1 - q^m$. Applying the generalization bound in Agarwal & Niyogi (2009) gives:

$$\epsilon \leq q^m + \tilde{O}\left(\frac{B}{\lambda M q} + \left(\frac{B^2}{\lambda} + Z\right)\sqrt{\frac{\ln(1/\delta)}{Mq}}\right) \tag{14}$$

where $Z = \sup_S f(S)$ and $\lambda$ controls an additional regularizer. In Sec. 4 we use a stricter variant of this formulation, in which high-quantile items are binned separately.

### 3.5. Using PMAC algorithms in practice

In principle, the reduction in Sec. 2.1 shows that any PMAC algorithm can be used for DOPS. Practically, however, this approach has several disadvantages. The root cause of this is that most current PMAC algorithms are designed for general submodular functions.[2] As such, they must adhere to demanding lower bounds (Balcan & Harvey, 2011; Feldman & Vondrak, 2016) which hold even for simple distributions (e.g., uniform). When considering specific submodular subclasses, these algorithms can therefore be suboptimal (and in fact quite costly) in terms of runtime, sample complexity, and/or approximation ratio. Additionally, virtually all current PMAC algorithms provide guarantees for either uniform or product distributions. Even in this setting, PMAC algorithms either guarantee a fixed approximation ratio, or are exponential in $\alpha$ (Feldman & Vondrak, 2016), making them difficult to use for $\alpha$-DOPS with arbitrarily small $\alpha$. The only known result for arbitrary distributions is the $\sqrt{n+1}$-PMAC algorithm of Balcan & Harvey (2011), which give a matching $\tilde{\Omega}(n^{1/3})$ lower bound on $\alpha$.

## 4. Experiments

In this section we evaluate the performance of our method on the task of optimally choosing trending items in social media platforms. Of the countless items that are continuously created and shared by users in such platforms, only a handful will become widespread (Goel et al., 2012). A key

[2] A notable exception to this is Feldman & Kothari (2014) which specifically considers PMAC learning of coverage functions with unknown cover sets.

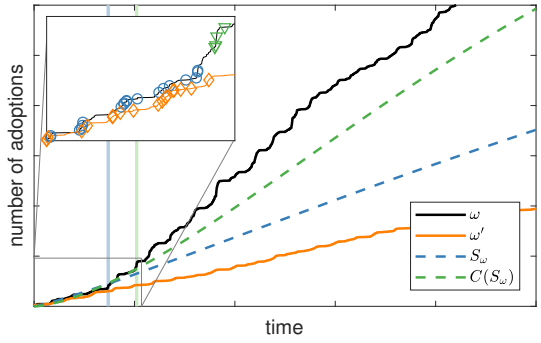


*Figure 1.* **Demonstrating the power of a coverage model.** The true diffusion curves of a focal hashtag $\omega$ (black) and an additional hashtag $\omega'$ with an initially similar (but eventually very different) diffusive pattern (orange). Diffusion-curve extrapolations (Bauckhage & Kersting, 2014) are generated based on $S_\omega$ alone (dashed blue) and on both $S_\omega$ and $C(S_\omega)$ (dashed green), with dashed lines marking the times of the corresponding last observations. This shows how conditioning on $C(S_\omega)$ can boost performance by providing a probabilistic "glimpse" into the near future. Markers in the zoomed inlaid plot indicate active users.

challenge faced daily by platform administrators is that of identifying potential trending content as early as possible. Trending items can then be marked, used for generating recommendations, or promoted to the public front page.

### 4.1. Optimizing trending items

For a given social platform, let $n$ be the number of users, and $\Omega$ be the set of spreading content items. When a user $u \in [n]$ is observed to have been exposed to an item $\omega \in \Omega$, we say that $u$ *adopted* $\omega$. This can happen, for instance, when $u$ views, shares, comments, or votes on $\omega$. A crucial factor in the successful spread of an item is the identity of its *early adopters* (Rogers, 1962; Goldenberg et al., 2002). We therefore represent each content item $\omega$ at a certain time point by the set of users that have adopted it up to that time, which we denote by $S_\omega \subseteq [n]$. We will be interested in the final number of adopters $z_\omega$ as a function of the set of adopting users, namely $z_\omega = f(S_\omega)$. For simplicity we assume that all items are considered at the time when adopted by exactly $k$ users, so that $|S_\omega| = k$ for all $\omega \in \Omega$. Under the above representation, targeting a successful item can be thought of as optimizing over the set of adopting users under a cardinality constraint. The task is therefore to choose the set $S_\omega$ for which $f(S_\omega)$ is maximal.

The above optimization task has two clear restrictions. First, $f$ cannot be accessed or queried, and any information regarding the value of subsets is available only via samples, namely past items and their adopting users. Second, an algorithm cannot output any user subset $S \subseteq [n]$, but must rather choose from a set of currently available items. In addition, the task of choosing the top trending item is performed repeatedly, each time over a different collection of content items. For example, for a front page that is updated hourly, a new trending item must be selected from the set of currently propagating content items for each update. Note that in such systems, the available subsets and their eventual value are primarily determined by the system's users. Online social platforms are therefore a prime example of a setting where an optimization algorithm has only statistical access to data.

### 4.2. Experimental setup

We evaluate the performance of our method on a benchmark dataset of propagating Twitter hashtags (Weng et al., 2013). Data was gathered by monitoring the sharing (tweeting and retweeting) of hashtags across users over the course of a month. The dataset includes 612,355 users who shared 226,488 distinct hashtags, with a total of 1,687,704 sharing activities. For each hashtag, the data describes the sequence of adopting users and the corresponding timestamps. These are used to construct a "retweet" social network $G = (V, E)$ where $(u, v) \in E$ if $v$ retweeted $u$. A user is considered to be *active* if she shared at least 20 hashtags. We focus on the 11,815 active users and on the 4,155 hashtags that include at least one active user. If a user retweeted the same hashtag more than once, we consider only the first tweet.

Samples were generated in the following manner. For each hashtag $\omega$, the user set $S_\omega$ was defined to include the first $k \in \{5, \ldots, 15\}$ active adopting users, and $z_\omega$ was set to be the number of eventual adopters. All pairs $(S_\omega, z_\omega)$ were randomly partitioned into a train set $\mathcal{S}$ and a global test set $\mathcal{T}'$ using a 90:10 split. All methods were given $\mathcal{S}$ as input, and were evaluated on 1,000 random subsets $\mathcal{T} \subseteq \mathcal{T}'$ of size $m$, where $m \in \{100, \ldots, 500\}$. This was repeated 100 times, and average results are reported. All methods we consider return an element $\hat{S} \in \mathcal{T}$ by computing $\operatorname{argmax}_{S \in \mathcal{T}} g(S)$ for some score function $g$, which is typically learned from the data. Hyper-parameters were tuned using cross validation for all relevant methods.

**`DOPS` model:** We implement the `DOPS` algorithm using coverage functions as the base class. Specifically, given the social network graph $G = (V, E)$, we use $V$ as the ground set, and construct a cover set $C_v = u : (v, u) \in E)$ for every $v \in V$. The coverage function we learn is:

$$f_{\theta,\eta}(S) = \sum_{v \in V} \theta_v + \sum_{u \in C(S)} \eta_u \qquad (15)$$

where $C(S) = \bigcup_{v \in S} C_v$. The idea behind this model is that, given that user $v$ adopted, each of her neighbors can also adopt (with some probability). Figure 1 illustrates this idea. Thus, the two terms in Eq. (15) quantify the contributions of the adopting nodes and of their neighbors, respectfully,

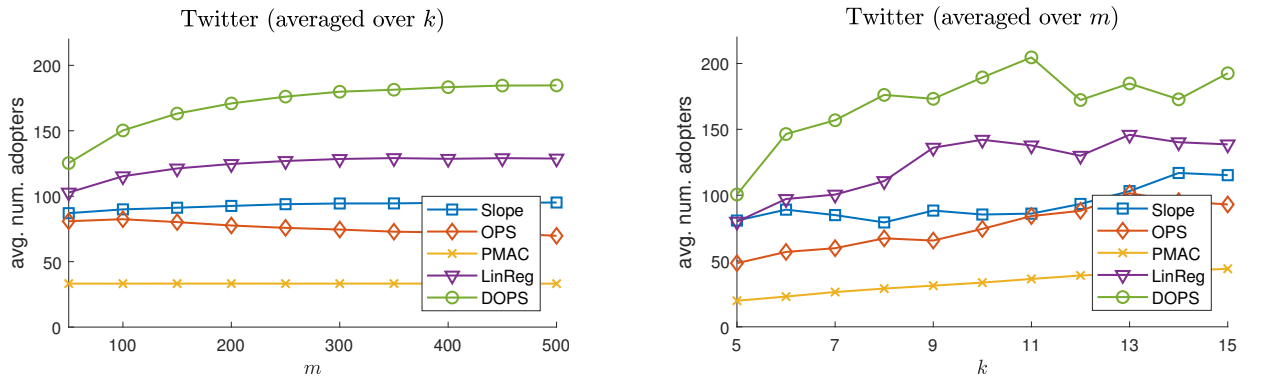


*Figure 2.* Comparison different methods for the task of optimally choosing trending hashtags on Twitter.

to the overall score. The coverage formulation takes into account the potential overlap in neighboring nodes, which can often be considerable (Holland & Leinhardt, 1971; Watts & Strogatz, 1998). We note that $G$ is constructed using training data alone, and incoming edges where only considered for nodes with at least 10 shares. Eq. (10) was optimized using the cutting-plane method of Joachims et al. (2009).

**Baselines:** We compare to the following methods:

- **SLOPE:** A first-order extrapolation where we first estimate the slope of the diffusion curve, and then choose the subset with the highest value.
- **LINREG:** We first run linear regression with $\ell_2$ regularization, and then choose the subset with the highest predicted value.
- **OPS:** A variant of the `OPS` (Balkanski et al., 2016), where instead of returning a global argmax, a given subset is scored based on the sum of marginal estimates. Note that under certain conditions, this algorithm is optimal for the setting of optimization from samples.
- **PMAC:** A soft version of the distribution-independent `PMAC` algorithm of Balcan & Harvey (2011). Since the original algorithm assumes separability (which does not hold here), we instead use an agnostic classifier.

**Results:** Figures 2(a) and 2(b) compare the value (number of adopters) for the chosen output of each method. As can be seen, `DOPS` clearly outperforms other methods by a margin. Note that when $k$ increases, average output values are likely to increase as well, since the algorithms are given more information as input. When $m$ increases, however, it is not clear a-priori how the average output values should change. This is because larger test sets are more likely to include higher-valued items, but at the same time have more low-valued alternatives. Interestingly, while the performance of most baselines does not improve (or even degrades) as $m$ increases, the performance of `DOPS` improves steadily.

## 5. Conclusions

In this work, we proposed an optimization criterion for settings where the algorithm is limited to statistical access of the objective function. We argue that this setting is pervasive, and in fact, believe that in most applications it is the common rule rather than the exception. Previous results have been generally negative, but mostly due to demanding worst-case requirements. Drawing inspiration from learning theory, our solution relaxes these requirements to hold in expectation.

Our main theoretical result shows an equivalence between optimization in this setting and learning. This highlights intriguing connections between the computational and statistical structure of function classes. An interesting corollary is that analyzing hardness of computation and approximation can now be done using statistical tools, and vice versa.

Several of the functions classes we explored are notoriously hard to optimize, but have a surprisingly simple structure as a function of their parameters. This allowed us to use simple learning strategies to produce powerful optimization mechanisms. We hypothesize that there are many other classes that posses these properties. An additional avenue for further exploration, hinted by our equivalence result, is the reverse: are there classes that are seemingly hard-to-learn, but due to their optimizational properties, can actually be learned efficiently? We leave this for future work.

## Acknowledgements

This research was supported by a Google PhD Fellowship, NSF grant CAREER CCF-1452961, BSF grant 2014389, NSF USICCS proposal 1540428, ISF Centers of Excellence grant, a Google research award, and a Facebook research award.

## References


Agarwal, Shivani and Niyogi, Partha. Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research*, 10(Feb):441–474, 2009.

Balcan, Maria-Florina and Harvey, Nicholas JA. Learning submodular functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 793–802. ACM, 2011.

Balkanski, Eric, Rubinstein, Aviad, and Singer, Yaron. The power of optimization from samples. In *Advances in Neural Information Processing Systems*, pp. 4017–4025, 2016.

Balkanski, Eric, Rubinstein, Aviad, and Singer, Yaron. The limitations of optimization from samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pp. 1016–1027, 2017.

Bauckhage, Christian and Kersting, Kristian. Strong regularities in growth and decline of popularity of social media services. *arXiv preprint arXiv:1406.6529*, 2014.

Collins, Michael. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. *New developments in parsing technology*, 23:19–55, 2004.

Defazio, Aaron and Caetano, Tiberio S. A convex formulation for learning scale-free networks via submodular relaxation. In *Advances in Neural Information Processing Systems*, pp. 1250–1258, 2012.

Dughmi, Shaddin and Vondrák, Jan. Limitations of randomized mechanisms for combinatorial auctions. *Games and Economic Behavior*, 92:370–400, 2015.

Feldman, Vitaly and Kothari, Pravesh. Learning coverage functions and private release of marginals. In *Conference on Learning Theory*, pp. 679–702, 2014.

Feldman, Vitaly and Vondrak, Jan. Optimal bounds on approximation of submodular and XOS functions by juntas. *SIAM Journal on Computing*, 45(3):1129–1170, 2016.

Goel, Sharad, Watts, Duncan J, and Goldstein, Daniel G. The structure of online diffusion networks. In *Proceedings of the 13th ACM conference on electronic commerce*, pp. 623–638. ACM, 2012.

Goldenberg, Jacob, Libai, Barak, and Muller, Eitan. Riding the saddle: How cross-market communications can create a major slump in sales. *Journal of Marketing*, 66(2):1–16, 2002.

Golovin, Daniel and Krause, Andreas. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Gomes, Ryan and Krause, Andreas. Budgeted nonparametric learning from data streams. In *ICML*, pp. 391–398, 2010.

Gomez Rodriguez, Manuel, Leskovec, Jure, and Krause, Andreas. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1019–1028. ACM, 2010.

Guillory, Andrew and Bilmes, Jeff A. Simultaneous learning and covering with adversarial noise. In *ICML*, volume 11, pp. 369–376, 2011.

Hoi, Steven CH, Jin, Rong, Zhu, Jianke, and Lyu, Michael R. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pp. 417–424. ACM, 2006.

Holland, Paul W and Leinhardt, Samuel. Transitivity in structural models of small groups. *Comparative Group Studies*, 2(2):107–124, 1971.

Iyer, Rishabh K, Jegelka, Stefanie, and Bilmes, Jeff A. Curvature and optimal algorithms for learning and minimizing submodular functions. In *Advances in Neural Information Processing Systems*, pp. 2742–2750, 2013.

Joachims, T., Finley, T., and Yu, Chun-Nam. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1): 27–59, 2009.

Kempe, David, Kleinberg, Jon, and Tardos, Éva. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146. ACM, 2003.

Krause, Andreas and Golovin, Daniel. Submodular function maximization., 2014.

Lapin, Maksim, Hein, Matthias, and Schiele, Bernt. Top-k multiclass svm. In *Advances in Neural Information Processing Systems*, pp. 325–333, 2015.

Lin, Hui and Bilmes, Jeff. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 510–520. Association for Computational Linguistics, 2011.

Rodriguez, Manuel Gomez and Schölkopf, Bernhard. Submodular inference of diffusion networks from multiple trees. *arXiv preprint arXiv:1205.1671*, 2012.

Rogers, E.M. *Diffusion of innovations*. Free Press of Glencoe, 1962.

Sipos, Ruben, Shivaswamy, Pannaga, and Joachims, Thorsten. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 224–233. Association for Computational Linguistics, 2012.

Valiant, Leslie G. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Watts, Duncan J and Strogatz, Steven H. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440, 1998.

Weng, Lilian, Menczer, Filippo, and Ahn, Yong-Yeol. Virality prediction and community structure in social networks. *Scientific reports*, 3:2522, 2013.

# Appendix

## A. Missing Analysis from Section 2

**Theorem 5.** *For all $\alpha > 1$ and $\epsilon, \delta > 0$, for $m$ sufficiently large, there exists a family of functions $\mathcal{F}$ and a function $M_{\texttt{PMAC}}(\cdot)$ such that*

- *for all $\epsilon', \delta' > 0$: $\mathcal{F}$ is $\alpha$-PMAC-learnable with sample complexity $M_{\texttt{PMAC}}(n, \delta', \epsilon', \alpha)$, and*
- *given strictly less than $M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha)$ samples, $\mathcal{F}$ is not $\alpha$-DOPS, i.e.,*

$$M_{\texttt{DOPS}}(n, m, \delta, \epsilon, \alpha) \geq M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha).$$

*Proof.* Fix $\alpha > 1$ and $\epsilon > 0$. Define $p := 1-(1-\epsilon)^{1/m}+\epsilon_s$, for some small constant $\epsilon_s > 0$, and let $S_1, \ldots, S_{1/p}$ be $1/p$ arbitrary distinct sets. The hard class of functions is $\mathcal{F} = \{f_i\}_{i \in [1/p]}$ where

$$f_i(S) = \begin{cases} \alpha & \text{if } S = S_i \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

Consider the distribution $\mathcal{D}$ which is the uniform distribution over sets $S_1, \ldots, S_{1/p}$, so $S_j$ is drawn with probability $p$ for all $j \in [1/p]$. We first argue that the sample complexity for PMAC-learning $f$ over $\mathcal{D}$ is at most

$$M_{\texttt{PMAC}}(n, \delta', \epsilon', \alpha) = \begin{cases} 0 & \text{if } \epsilon' \geq p \\ \frac{\log(1/\delta')}{\log(1/(1-p))} & \text{if } \epsilon' < p \end{cases}$$

Note that if $\epsilon' \geq p$, $\tilde{f}(S) = 1/2$ for all $S$ is correct with probability $1 - p \geq 1 - \epsilon'$ over $S \sim \mathcal{D}$ and with probability 1 over the samples. If $\epsilon' < p$, if there exists sample $S_i$ such that $f(S_i) = \alpha$, then $\tilde{f}(S_i) = \alpha$, and $\tilde{f}(S) = 1/2$ for all other $S$. Note that that this is correct with probability 1 over $S \sim \mathcal{D}$ if $S_i$ is in the samples. The probability that $S_i$ is in the samples

$$\begin{aligned} 1-(1-p)^m &= 1-(1-p)^{\frac{\log(1/\delta')}{\log(1/(1-p))}} \\ &= 1-e^{\frac{\log(\delta')}{\log(1-p)}\log(1-p)} \\ &= 1-\delta'. \end{aligned}$$

Thus, $\tilde{f}$ is correct with probability $1-\delta'$ over the samples. Next, we argue that for all $\delta > 0$ and $m$ sufficiently large, the sample complexity for DOPS is at least

$$\begin{aligned} &M_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha) = \\ &M_{\texttt{PMAC}}(n, \delta, p-\epsilon_s, \alpha) = \frac{\log(1/\delta)}{\log(1/(1-p))}. \end{aligned}$$

Consider the random function $f_i$ where $i \in [1/p]$ is uniformly random. Let $\mathcal{F}'$ be the randomized collection of functions $f_i$ such that $S_i$ is in the testing set but not in the training set. Since $S_i$ is not in the testing set, we have that for all $f_i \in \mathcal{F}'$ and for all sets $S$ in the testing set,

$$f_i(S) = 1.$$

Thus, the functions in $\mathcal{F}'$ are *indistinguishable* from the samples in the training set. This implies that the decisions of the algorithm are *independent* of the random variable $i$, *conditioned* on $f_i \in \mathcal{F}'$. Let $S$ be the set in the testing set that is returned by the algorithm, we obtain that

$$\begin{aligned} \mathop{\mathbb{E}}_{i:f_i \in \mathcal{F}'}[f_i(S)] &= \Pr_{i:f_i \in \mathcal{F}'}[S = S_i] \cdot \alpha + \Pr_{i:f_i \in \mathcal{F}'}[S \neq S_i] \cdot \frac{1}{2} \\ &\leq \frac{\alpha}{|\mathcal{F}'|} + \frac{1}{2} \end{aligned}$$

since $S$ is independent of $i$ conditioned on $f_i \in \mathcal{F}'$. Consider the case where $S_i$ is not in the training set with probability strictly greater than $\delta$. The probability that $S_i$ is in the testing set is $1-(1-p)^m = \epsilon + \epsilon_s$. Thus a function is in $\mathcal{F}'$ with probability at least $\delta(\epsilon + \epsilon_s)$. Note that $1/p$ is arbitrarily large if $m$ is arbitrarily large. Thus, $|\mathcal{F}'| > 2\alpha$ with arbitrarily large probability if $m$ is arbitrarily large for fixed $\epsilon, \delta$, and $\alpha$. Combining with the previous inequality, this implies that

$$\begin{aligned} \mathop{\mathbb{E}}_{i:f_i \in \mathcal{F}'}[f_i(S)] < 1 &= \frac{1}{\alpha} \cdot f_i(S_i) \\ &= \frac{1}{\alpha} \cdot \mathop{\mathbb{E}}_{i:f_i \in \mathcal{F}'}[\max_{S \in \mathcal{S}_{te}} f_i(S)] \end{aligned}$$

where the last equality is since $S_i \in \mathcal{S}_{te}$ for all $i \in \mathcal{F}'$. Thus, there exists at least one function $f_i \in \mathcal{F}$ such that the algorithm does not obtain an $\alpha$-approximation when $S_i$ is in the testing set and not in the training set.

The probability that $S_i$ is in the testing set is $1-(1-p)^m = \epsilon + \epsilon_s$. Thus, $S_i$ needs to be in the training set with probability at least $1-\delta$, otherwise we don't get an $\alpha$-apx with probability $1-\epsilon$. The probability that $S_i$ is not in the training set is $(1-p)^m$. Thus, we need $\delta > (1-p)^m$, or

$$\begin{aligned} m &> \frac{\log(1/\delta)}{\log(1/(1-p))} \\ &= m_{\texttt{PMAC}}(n, \delta, p-\epsilon_s, \alpha) \\ &= m_{\texttt{PMAC}}(n, \delta, 1-(1-\epsilon)^{1/m}, \alpha). \end{aligned}$$

□