

Learning Submodular Functions

Maria-Florina Balcan*

Nicholas J. A. Harvey†

Abstract

There has recently been significant interest in the machine learning community on understanding and using submodular functions. Despite this recent interest, little is known about submodular functions from a learning theory perspective. Motivated by applications such as pricing goods in economics, this paper considers PAC-style learning of submodular functions in a distributional setting.

A problem instance consists of a distribution on $\{0, 1\}^n$ and a real-valued function on $\{0, 1\}^n$ that is non-negative, monotone and submodular. We are given $\text{poly}(n)$ samples from this distribution, along with the values of the function at those sample points. The task is to approximate the value of the function to within a multiplicative factor at subsequent sample points drawn from the same distribution, with sufficiently high probability. We prove several results for this problem.

- If the function is Lipschitz and the distribution is a product distribution, such as the uniform distribution, then a good approximation is possible: there is an algorithm that approximates the function to within a factor $O(\log(1/\epsilon))$ on a set of measure $1 - \epsilon$, for any $\epsilon > 0$.
- If we do not assume that the distribution is a product distribution, then the approximation factor must be much worse: no algorithm can approximate the function to within a factor of $\tilde{O}(n^{1/3})$ on a set of measure $1/2 + \epsilon$, for any constant $\epsilon > 0$. This holds even if the function is Lipschitz.
- On the other hand, this negative result is nearly tight: for an arbitrary distribution, there is an algorithm that approximates the function to within a factor \sqrt{n} on a set of measure $1 - \epsilon$.

Our work combines central issues in optimization (submodular functions and matroids) with central topics in learning (distributional learning and PAC-style analyses) and with central concepts in pseudo-randomness (lossless expander graphs). Our analysis involves a twist on the usual learning theory models and uncovers some interesting structural and extremal properties of submodular functions, which we suspect are likely to be useful in other contexts. In particular, to prove our general lower bound, we use lossless expanders to construct a new family of matroids which can take wildly varying rank values on superpolynomially many sets; no such construction was previously known.

*Georgia Institute of Technology, School of Computer Science. Email: ninamf@cc.gatech.edu. Portions of this work were done while at Microsoft Research, New England.

†University of Waterloo, Department of Combinatorics and Optimization. Email: harvey@math.uwaterloo.ca. Portions of this work were done while at Microsoft Research, New England.

1 Introduction

What does it mean to “learn a submodular function”, and why would one be interested in doing that? To begin, let us explain what a submodular function is. Let $[n] = \{1, \dots, n\}$ be a ground set and let $f : 2^{[n]} \rightarrow \mathbb{R}$ be a set function. This function is called *submodular* if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad \forall A, B \subseteq [n]. \quad (1.1)$$

Submodularity is in many ways similar to concavity of functions defined on \mathbb{R}^n . For example, concavity of differentiable functions is equivalent to gradient monotonicity, and submodularity is equivalent to monotonicity of the marginal values:

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B) \quad \forall A \subseteq B \subseteq [n] \text{ and } i \notin B. \quad (1.2)$$

This inequality reflects a natural notion of “diminishing returns”, which explains why submodularity has long been a topic of interest in economics [56]. Submodular functions have also been studied for decades in operations research and combinatorial optimization [15], as they arise naturally in the study of graphs, matroids, covering problems, facility location problems, etc.

More recently, submodular functions have become key concepts in both the machine learning and algorithmic game theory communities. For example, submodular functions have been used to model bidders’ valuation functions in combinatorial auctions [26, 42, 14, 5, 60], for solving feature selection problems in graphical models [39], and for solving various clustering problems [47]. In fact, submodularity in machine learning has been a topic of two tutorials at two recent major conferences in machine learning [40, 41].

The Model. So what does it mean to “learn a submodular function”? Our definition comes from the field of computational learning theory, where the goal of learning is to make predictions about the future based on past observations. One of the most successful approaches to formalizing this goal is Valiant’s PAC model [58]; even today, this model continues to stimulate significant research in theoretical computer science. However, the PAC model has primarily been designed for learning *Boolean-valued functions*, such as threshold functions and low-depth circuits [58, 37]. For *real-valued functions*, it seems appropriate to change the model by ignoring small-magnitude errors in the predicted values. Our results on learning submodular functions are presented in this new model, which we call the *PMAC model*: this abbreviation stands for “Probably Mostly Approximately Correct”.

In this model, a learning algorithm is given a set \mathcal{S} of polynomially many labeled examples drawn i.i.d. from some fixed, but unknown, distribution D over points in $2^{[n]}$. The points are labeled by a fixed, but unknown, target function $f^* : 2^{[n]} \rightarrow \mathbb{R}_+$. The goal is to output a hypothesis function f such that, with large probability over the choice of examples, the set of points for which f is a good approximation for f^* has large measure with respect to D . More formally,

$$\Pr_{x_1, x_2, \dots \sim D} [\Pr_{x \sim D} [f(x) \leq f^*(x) \leq \alpha f(x)] \geq 1 - \epsilon] \geq 1 - \delta,$$

where f is the output of the learning algorithm when given inputs $\{(x_i, f^*(x_i))\}_{i=1,2,\dots}$ and the approximation ratio $\alpha \geq 1$ allows for multiplicative error in the function values. In our model, one must *approximate* the value of a function on a set of large measure, with high confidence. In contrast, the traditional PAC model requires one to predict the value *exactly* on a set of large measure, with high confidence. The PAC model is the special case of our model with $\alpha = 1$.

Motivation. So why would one want to learn a submodular function in this manner? Our work has multiple motivations. From a foundational perspective, submodular functions form a broad class of important functions, so studying their learnability allows us to understand their structure in a new way. To draw a parallel to the Boolean-valued case, a class of comparable breadth is the class of monotone Boolean functions, which have been intensively studied [9, 7, 2, 12, 48].

From an applications perspective, algorithms for learning submodular functions could be very useful in some of the applications where these functions arise. A classic example is pricing bundles of goods. For example, a software company which produces a software suite typically produces several bundles, each of which is a subset of the software programs in the suite. Further examples abound: automobile manufacturers produce a few trim lines of their vehicles with various added options; supermarkets sell bundles of condiments, variety packs of cereal, etc. From an economic standpoint, the central question here is *bundle pricing*: how should a company decide which bundles to sell, and how should they choose their prices? This is an active area of research in management science and microeconomic theory [26, 62, 4, 50, 18].

There has been much work on methods for designing optimally priced bundles, in various models. These methods typically make two assumptions [26]. First, the consumer’s valuations for all possible bundles are known. Second, the consumer’s valuations exhibit economies of scale (e.g., subadditivity or submodularity). Our work is motivated by the observation that this first assumption is entirely unrealistic, both computationally and pragmatically, since the number of bundles is exponential in the number of goods.

We propose learning of submodular functions as an approach to make this first assumption more realistic. To justify our proposal, note that the that corporations typically possess large amounts of data acquired from past consumer purchases [50]. This suggests that *passive supervised learning* is a realistic model for learning consumer valuations. Next, we note that valuations are real numbers, so it may not be possible to learn them exactly. This suggests that allowing solutions with *multiplicative error* is appropriate for this problem. Thus, the problem of learning submodular functions in the PMAC model is a good fit for the real-world problem of learning consumer valuations.

1.1 Overview of Our Results and Techniques

In this work, we prove several algorithmic results and lower bounds in the PMAC model, as well as new surprising structural results concerning submodular functions and matroids. To do so, we combine ideas and techniques from computational learning theory, combinatorial optimization, pseudorandomness and discrete probability theory.

Algorithm for product distributions. We begin with a positive result. We show that non-negative, monotone, submodular functions can be PMAC-learned with a constant approximation factor α , under the additional assumptions that the distribution D on examples is a product distribution, and that the function is Lipschitz. Focusing on product distributions is quite a natural restriction — in learning theory, it is common to study learnability of functions under the uniform distribution or a product distribution [43, 38, 35], particularly when the class of functions has high complexity.

The main technical result underlying this algorithm is a concentration result for monotone, submodular, Lipschitz functions. Using Talagrand’s inequality, we show that such functions are extremely tightly concentrated around their expected value. Therefore the submodular function is actually well-approximated by the *constant* function that equals the empirical average on all points.

Inapproximability for general distributions. Given the simplicity of the constant-approximation algorithm for product distributions, a natural next step would be to generalize it to arbitrary distributions. Rather surprisingly, we show that this is impossible: under arbitrary distributions, every algorithm for PMAC-learning monotone, submodular functions must have approximation factor $\tilde{\Omega}(n^{1/3})$, even if the functions are Lipschitz. Moreover, this lower bound holds even if the algorithm knows the underlying distribution and it can adaptively query the target function at points of its choice.

This inapproximability result is the most technical part of our paper. To prove it, we require a family of submodular functions which take wildly varying values on a certain set of points. If the target function is drawn from this family, then the learning algorithm will not be able to predict the function values on those points, and therefore must have a high approximation ratio. We obtain such a family of submodular

functions by creating a new family of matroid with surprising extremal properties, which we describe below.

Algorithm for general distributions. Our $\tilde{\Omega}(n^{1/3})$ inapproximability result for general distributions is dire, but it turns out to be nearly optimal. We give an algorithm to PMAC-learn an arbitrary non-negative, monotone, submodular function with approximation factor $O(\sqrt{n})$. The approximation ratio of this algorithm is large, but not unreasonable — it has recently emerged that $n^{\Theta(1)}$ is the correct approximation ratio for numerous optimization problems involving submodular functions [21, 54, 20, 33].

This algorithm is based on a recent structural result which shows that any monotone, non-negative, submodular function can be approximated within a factor of \sqrt{n} on every point by the square root of a linear function [21]. We show how we can leverage this result to reduce the problem of PMAC-learning a submodular function to learning a linear separator in the usual PAC model. We remark that an improved structural result for any subclass of submodular functions immediately implies an improved analysis of our algorithm for that subclass.

A family of extremal matroids. Our inapproximability result is based on a new family of matroids¹ with several interesting properties. The technical question we explore is: given a set family $\mathcal{A} = \{A_1, \dots, A_k\}$ and $b_1, \dots, b_k \in \mathbb{Z}$, when is

$$\mathcal{I} = \{ I : |I| \leq r \wedge |I \cap A_i| \leq b_i \ \forall i = 1, \dots, k \} \tag{1.3}$$

a matroid? The simplest matroid of this type is obtained when the A_i 's are disjoint, in which case \mathcal{I} is a (truncated) partition matroid. Another important matroid of this type is obtained when the A_i 's form an error-correcting code of constant weight r and minimum distance 4, and each $b_i = r - 1$; such a matroid is a *paving matroid*.

To this date, there has been no unified explanation for these two types of matroids. Our observation is that these two special cases are matroids due to the *expansion* of the set system \mathcal{A} : disjoint sets expand perfectly, and error-correcting codes are precisely *pairwise* expanders. This suggests the general question: if \mathcal{A} has good expansion properties, does \mathcal{I} form a matroid? For example, if the A_i 's are almost disjoint, can we obtain a matroid that's almost a partition matroid? We give a positive answer to these questions, subject to some additional technical conditions. This matroid construction, together with the existence of expanders with certain parameters, and the fact that the b_i 's can be (almost) arbitrary, gives a family of matroids taking wildly varying rank values on the A_i 's. This leads to our $\tilde{\Omega}(n^{1/3})$ inapproximability result.

Approximate characterization of matroids. Our final result is an interesting “approximate characterization” of matroids. It is well-known that defining the function $f : 2^{[n]} \rightarrow \mathbb{R}$ by $f(S) = h(|S|)$ gives a submodular function if $h : \mathbb{R} \rightarrow \mathbb{R}$ is concave. Surprisingly, we show that an approximate converse is true: for any matroid, there exists a concave function h such that most sets S have rank approximately $h(|S|)$. A more precise statement is in Section 3.1. This result is also based on our concentration inequality for Lipschitz, submodular functions under product distributions.

1.2 Related Work

Submodular Functions. Optimization problems involving submodular functions has been a highly active topic over the past few years. There has been significant progress on algorithms (or approximation algorithms) for both minimizing and maximizing submodular functions [22, 51, 29, 31, 17], under various constraints. Approximation algorithms for submodular analogues of several other well-known optimization problems have been studied, including load balancing [54], set cover [63, 30], shortest path [20], sparsest cut [54], (s, t) -cut [33], vertex cover [30, 20], etc.

Learning real-valued functions and the PMAC Model. In the machine learning literature [27, 59], learning real-valued functions (in the usual distributional learning setting) is often addressed by considering the

¹ For a brief definition of matroids, see Section 2. For further discussion, we refer the reader to standard references [49, 52].

squared error loss², i.e. $\mathbf{E}_x [(f(x) - f^*(x))^2]$. The squared error loss, however, does not distinguish between the case of having low error on most of the distribution and just high error over a few points, versus having moderately high error everywhere. Thus, for instance, a lower bound for the squared error loss is not so meaningful. Instead, the PMAC model allows for more fine-grained control with separate parameters for the amount and extent of errors.

Learning Submodular Functions. To our knowledge, there is no prior work on learning submodular functions in a natural distributional PAC style learning setting. The most relevant work is a paper of Goemans et al. [21], which considers the problem of “approximating submodular functions everywhere”. That paper considers the algorithmic problem of efficiently finding a function which approximates a submodular function at every point of its domain. They give an algorithm which achieves an approximation factor $\tilde{O}(\sqrt{n})$, and also show $\tilde{\Omega}(\sqrt{n})$ inapproximability. Their algorithm adaptively queries the target function at points of its choice, and the hypothesis it produces must approximate the target function at *every* point.³ In contrast, our learnability results (within the PMAC model) are learning results for the more widely studied passive supervised learning setting [3, 37, 58, 59], which is more relevant for our motivating application to bundle pricing.

Our algorithm for PMAC-learning under general distributions and the Goemans et al. algorithm both rely on the structural result (due to Goemans et al.) that monotone, submodular functions can be approximated by the square root of a linear function to within a factor \sqrt{n} . In both cases, the challenge is to find this linear function. The Goemans et al. algorithm is very sophisticated: it gives an intricate combinatorial algorithm to approximately solve a certain convex program which produces the desired function. On the other hand, our algorithm is very simple: given the structural result, we can reduce our problem to that of learning a linear separator, which is easily solved by linear programming. Moreover, our algorithm is noise-tolerant and more amenable to extensions; we elaborate on this in Section 5 and Appendix G.1.

On the other hand, our lower bound is significantly more involved than lower bound of Goemans et al. Both of these lower bounds involve matroids of the form in Eq. (1.3), although Goemans et al. only need such matroids for the easy case $k = 1$. Handling the case $k = n^{\omega(1)}$ makes our matroid construction much more intricate. Essentially, Goemans et al. only show worst-case inapproximability, whereas we need to show *average-case inapproximability*. The situation is similar with Boolean functions, where lower bounds for distributional learning are typically much harder to show than lower bounds for exact learning. For instance, even conjunctions are hard to learn in the exact learning model, and yet it’s trivial to PAC-learn them. Proving a lower bound for PAC-learning requires exhibiting some fundamental complexity in the class of target functions, especially when one does not restrict the form of the hypothesis function. It is precisely this phenomenon which makes our lower bound challenging to prove.

2 Formalizing the Model

2.1 Preliminaries

Notation. Throughout this paper, we let $[n]$ denote the set $\{1, 2, \dots, n\}$. This will typically be used as the ground set for the matroids and submodular functions that we discuss. For any set $S \subseteq [n]$ and element $x \in [n]$, we let $S + x$ denote $S \cup \{x\}$. The indicator vector of a set $S \subseteq [n]$ is $\chi(S) \in \mathbb{R}^n$, where $\chi(S)_i$ is 1 if i is in S and 0 otherwise.

²Other loss functions are also used, such as L_1 loss, but from our perspective they are not substantially different from squared error loss.

³Technically speaking, their model could be viewed as the “exact learning with value queries model”, which is not the most natural from a machine learning perspective. In particular, it is well-known that in many learning applications it is undesirable to allow arbitrary membership or value queries because natural oracles, such as hired humans, have difficulty labeling synthetic examples [6]. In addition, negative results for exact learning do not necessarily imply hardness for learning in other more widely used learning models and we discuss this in more detail below.

Recall that a *product distribution* on the set $2^{[n]}$ is a distribution D such that, if $S \subseteq [n]$ is a sample from D , then the events $i \in S$ and $j \in S$ are independent for every $i \neq j$.

Submodular Functions and Matroids. We now briefly state some standard facts about matroids and submodular functions. For a detailed discussion, we refer the reader to standard references [19, 44, 49, 52]. We will be concerned with the following properties of set functions. We say that $f : 2^{[n]} \rightarrow \mathbb{R}$ is

- *Normalized* if $f(\emptyset) = 0$.
- *Non-negative* if $f(S) \geq 0$ for all S .
- *Monotone* (or *non-decreasing*) if $f(S) \leq f(T)$ whenever $S \subseteq T$.
- *Submodular* if it satisfies Eq. (1.1), or equivalently Eq. (1.2).
- *Subadditive* if $f(S) + f(T) \geq f(S \cup T)$ for all $S, T \subseteq [n]$.
- *L-Lipschitz* if $|f(S+x) - f(S)| \leq L$ for all $S \subseteq [n]$ and $x \in [n]$.

Throughout this paper we will implicitly assume that all set functions are normalized. Some basic facts about submodular functions are given in Appendix A.

One manner in which submodular functions arise is as the rank functions of matroids. A pair $\mathbf{M} = ([n], \mathcal{I})$ is called a matroid if $\mathcal{I} \subseteq 2^{[n]}$ is a non-empty family such that

- if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$, and
- if $I, J \in \mathcal{I}$ and $|J| < |I|$, then there exists an $i \in I \setminus J$ such that $J + i \in \mathcal{I}$.

The sets in \mathcal{I} are called *independent* and those not in \mathcal{I} are called *dependent*. A maximal independent set is called a *base* of \mathbf{M} . All bases have the same size, which is called the *rank* of the matroid and is denoted $\text{rk}(\mathbf{M})$. The *rank function* of the matroid is the function $\text{rank}_{\mathbf{M}} : 2^{[n]} \rightarrow \mathbb{N}_+$ defined by

$$\text{rank}_{\mathbf{M}}(S) := \max \{ |I| : I \subseteq S, I \in \mathcal{I} \}.$$

It is well-known that $\text{rank}_{\mathbf{M}}$ is non-negative, monotone, submodular, and 1-Lipschitz.

2.2 The PMAC Model

The PMAC model is a passive, supervised learning framework. There is a space $\{0, 1\}^n$ of examples, and a fixed but unknown distribution D on $\{0, 1\}^n$. The examples are labeled by a fixed but unknown target function $f^* : \{0, 1\}^n \rightarrow \mathbb{R}_+$. In this model, a learning algorithm is provided a set \mathcal{S} of labeled training examples drawn i.i.d. from D and labeled by f^* . The algorithm may perform an arbitrary computation on the labeled examples \mathcal{S} , then must output a hypothesis function $f : \{0, 1\}^n \rightarrow \mathbb{R}_+$. The goal is that, with high probability, f is a good approximation of the target for most points in D . Formally, we define the PMAC learning model as follows.

Definition 1. Let \mathcal{F} be a family of non-negative, real-valued functions with domain $\{0, 1\}^n$. We say that an algorithm \mathcal{A} PMAC-learns \mathcal{F} with approximation factor α if, for any distribution D over $\{0, 1\}^n$, for any target function $f^* \in \mathcal{F}$, and for $\epsilon \geq 0$ and $\delta \geq 0$ sufficiently small:

- The input to \mathcal{A} is a sequence of pairs $\{(x_i, f^*(x_i))\}_{1 \leq i \leq \ell}$ where each x_i is chosen independently from distribution D .
- The number of inputs ℓ provided to \mathcal{A} and the running time of \mathcal{A} are both at most $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$.
- The output of \mathcal{A} is a function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ that satisfies

$$\Pr_{x_1, \dots, x_\ell \sim D} \left[\Pr_{x \sim D} [f(x) \leq f^*(x) \leq \alpha \cdot f(x)] \geq 1 - \epsilon \right] \geq 1 - \delta.$$

The name PMAC stands for ‘‘Probably Mostly Approximately Correct’’. It is an extension of the PAC model to learning non-negative, real-valued functions, allowing multiplicative error α . The PAC model for learning boolean functions is precisely the special case when $\alpha = 1$.

Learning submodular functions in the PMAC model. In this paper we focus on the PMAC-learnability of submodular functions. We note that it is quite easy to PAC-learn the class of *boolean* submodular functions.

Algorithm 1 An algorithm for PMAC-learning a non-negative, monotone, 1-Lipschitz, submodular function f^* when the examples come from a product distribution. Its input is a sequence of labeled training examples $(S_1, f^*(S_1)), \dots, (S_\ell, f^*(S_\ell))$, where $\ell = 10n^2 \log(1/\delta) + n \log(n/\delta)/\epsilon$.

- Let $\mu = \sum_{i=1}^{\ell} f^*(S_i)/\ell$.
 - *Case 1:* If $\mu \geq 450 \log(1/\epsilon)$, then return the constant function $f = \mu/4$.
 - *Case 2:* If $\mu < 450 \log(1/\epsilon)$, then compute the set $U = \bigcup_{i: f^*(S_i)=0} S_i$. Return the function f where $f(A) = 0$ if $A \subseteq U$ and $f(A) = \mu$ otherwise.
-

Details are given in Appendix B. The rest of this paper considers the much more challenging task of PMAC-learning the general class of real-valued, submodular functions. We also provide PMAC-learnability results for the more general class of subadditive functions.

3 An Algorithm for Lipschitz Functions and Product Distributions

In computational learning theory, it is common to study learnability of functions when the examples are distributed according to the uniform distribution or a product distribution [43, 38, 35]. In this section, we consider the learnability of submodular functions under such distributions. We show that Algorithm 1 PMAC-learns submodular functions with a constant approximation factor, under the additional assumption that the function is Lipschitz. Formally, our result is:

Theorem 1. *Let \mathcal{F} be the class of non-negative, monotone, 1-Lipschitz, submodular functions with ground set $[n]$ and minimum non-zero value η . Let D be a product distribution on $[n]$. For any sufficiently small $\epsilon > 0$ and $\delta > 0$, Algorithm 1 PMAC-learns \mathcal{F} with approximation factor $O(\log(1/\epsilon)/\eta)$. The number of training examples used is $10n^2 \log(1/\delta) + n \log(n/\delta)/\epsilon$.*

A detailed proof is in Appendix C; we now briefly describe the idea. We show that, under a product distribution, the value of f is tightly concentrated around its expectation. Consequently, the empirical average gives a good approximation of f for most of the distribution. Therefore f is well-approximated by the constant function that equals the empirical average on all points. This idea is employed in Case 1 of Algorithm 1.

One caveat is that allowing multiplicative error is of no help in estimating the zeros of f . The zeros must be treated specially. Fortunately the zeros of a non-negative, monotone, submodular function have special structure: they are both union-closed and downward-closed. In other words, the indicator function for the zeros is a NOR function. Therefore Case 2 handles the zeros by PAC-learning this NOR function.

The main technical ingredient in the proof of Theorem 1 is our concentration bound on f , which we state as Theorem 2.

Theorem 2. *Let $f : 2^{[n]} \rightarrow \mathbb{R}_+$ be a non-negative, monotone, submodular, 1-Lipschitz function. Let the random variable $X \subseteq [n]$ have a product distribution. For any $b, t \geq 0$,*

$$\Pr \left[f(X) \leq b - t\sqrt{b} \right] \cdot \Pr \left[f(X) \geq b \right] \leq \exp(-t^2/4). \quad (3.1)$$

To understand Theorem 2, it is instructive to compare it with known results. For example, the Chernoff bound is precisely a concentration bound for *linear*, Lipschitz functions. On the other hand, if f is an arbitrary 1-Lipschitz function then Azuma's inequality implies concentration, although of a much weaker form, with standard deviation roughly \sqrt{n} . So Theorem 2 can be viewed as saying that Azuma's inequality can be significantly strengthened when the given function is known to be submodular.

Our proof of Theorem 2 is based on the Talagrand inequality [55, 1, 46, 32]. Independently, Chekuri and Vondrák [11] proved a similar result using the FKG inequality. Concentration results of this flavor can also

be proven using the framework of self-bounding functions [8], as observed in an earlier paper by Hajiaghayi et al. [25]; see also the survey by Vondrák [61].

Theorem 2 most naturally implies concentration around the median of $f(X)$. By standard manipulations, e.g., [32, §2.5] or [46, §20.2], this also implies concentration around the expected value. We obtain:

Corollary 3. *Let $f : 2^{[n]} \rightarrow \mathbb{R}_+$ be a non-negative, monotone, submodular, 1-Lipschitz function. Let the random variable $X \subseteq [n]$ have a product distribution. For any $0 \leq \alpha \leq 1$ and if $\mathbf{E}[f(X)] \geq 240/\alpha$, then*

$$\Pr[|f(X) - \mathbf{E}[f(X)]| > \alpha \mathbf{E}[f(X)]] \leq 4 \exp(-\alpha^2 \mathbf{E}[f(X)]/16).$$

3.1 An Approximate Characterization of Matroid Rank Functions

We now present an ancillary result that is an application of the ideas in the previous section. The statement is somewhat surprising: matroid rank functions are very well approximated by *univariate*, concave functions. The proof is also based on Theorem 2. To motivate the result, consider the following easy construction of submodular functions, which can be found in Lovász’s survey [44, pp. 251]

Proposition 4. *Let $h : \mathbb{R} \rightarrow \mathbb{R}$ be concave. Then $f : 2^{[n]} \rightarrow \mathbb{R}$ defined by $f(S) = h(|S|)$ is submodular.*

It goes without saying that this construction yields an extremely restricted class of submodular functions. However, we now show that a partial converse is true.

Theorem 5. *Let $f : 2^{[n]} \rightarrow \mathbb{Z}_+$ be the rank function of a matroid with no loops, i.e., $f(S) \geq 1$ whenever $S \neq \emptyset$. Fix $\epsilon > 0$, sufficiently small. There exists a concave function $h : [0, n] \rightarrow \mathbb{R}$ such that, for every $k \in [n]$, and for a $1 - \epsilon$ fraction of the sets $S \in \binom{[n]}{k}$, we have*

$$\frac{1}{O(\log(1/\epsilon))} h(k) \leq f(S) \leq O(\log(1/\epsilon)) h(k).$$

The idea behind this theorem is as follows. We define $h(p)$ to be the expected value of f under the product distribution which samples elements with probability p . The value of f under this distribution is tightly concentrated around $h(p)$, by the results of the previous section. But the distribution defining $h(k/n)$ is very similar to the uniform distribution on sets of size k , so f is also tightly concentrated under the latter distribution. So the value of f for most sets of size k is roughly $h(k/n)$. The concavity of this function h is a consequence of submodularity of f . A detailed proof is in Appendix D.

4 Inapproximability under Arbitrary Distributions

The simplicity of Algorithm 1 might make one hope that a constant-factor approximation is possible under arbitrary distributions. Any such hopes are dashed by the following theorem.

Theorem 6. *No algorithm can PMAC-learn the class of non-negative, monotone, submodular functions with approximation factor $o(n^{1/3}/\log n)$.*

This result holds even if the algorithm is told the underlying distribution, even if the algorithm can query the function on inputs of its choice, and even if the queries are adaptive. In other words, this inapproximability still holds in the PMAC model augmented with value queries.

Theorem 6 is an information-theoretic hardness result. A slight modification yields Corollary 7, which is a complexity-theoretic hardness result. Proofs of Theorem 6 and Corollary 7 are given in Appendix E.2.

Corollary 7. *Suppose that one-way functions exist. For any constant $\epsilon > 0$, no algorithm can PMAC-learn the class of non-negative, monotone, submodular functions with approximation factor $O(n^{1/3-\epsilon})$, even if the functions are given via polynomial-time algorithms which compute their value on the support of the distribution.*

As we described in Section 1.1, we will prove these results by constructing a family of matroids whose rank functions take wildly varying values on a certain set of points. The following theorem gives this

construction. It is proven in Section 4.3.

Theorem 8. *For any $k = 2^{o(n^{1/3})}$, there exists a family of sets $\mathcal{A} \subseteq 2^{[n]}$ and a family of matroids $\mathcal{M} = \{ \mathbf{M}_{\mathcal{B}} : \mathcal{B} \subseteq \mathcal{A} \}$ with the following properties.*

- $|\mathcal{A}| = k$ and $|A| = n^{1/3}$ for every $A \in \mathcal{A}$.
- For every $\mathcal{B} \subseteq \mathcal{A}$ and every $A \in \mathcal{A}$, we have

$$\text{rank}_{\mathbf{M}_{\mathcal{B}}}(A) = \begin{cases} 8 \log k & (\text{if } A \in \mathcal{B}) \\ |A| & (\text{if } A \in \mathcal{A} \setminus \mathcal{B}). \end{cases}$$

To help understand how this theorem relates to Theorem 6, let us focus on the case where k is slightly super-polynomial, say $k = n^{\log n}$. In the matroid $\mathbf{M}_{\mathcal{B}}$, a set A has rank only $O(\log^2 n)$ if $A \in \mathcal{B}$, but has rank $\Omega(n^{1/3})$ if $A \in \mathcal{A} \setminus \mathcal{B}$. In other words, as \mathcal{B} varies, the rank of a set $A \in \mathcal{A}$ varies wildly, depending on whether $A \in \mathcal{B}$ or not. Thus, given an unknown matroid $\mathbf{M}_{\mathcal{B}} \in \mathcal{M}$, the problem of learning the rank of each $A \in \mathcal{A}$ amounts to learning the set $\mathcal{B} \subseteq \mathcal{A}$. This is not possible since $|\mathcal{A}|$ is super-polynomial, and this leads to the claimed $\tilde{\Omega}(n^{1/3})$ inapproximability for PMAC-learning these rank functions.

4.1 Discussion of Theorem 8

Another motivation for Theorem 8 is its connection to the *submodular completion problem*. Suppose we have partially defined a set function $f : 2^{[n]} \rightarrow \mathbb{R}$, perhaps assigning $f(A_1) = b_1$, $f(A_2) = b_2$, etc. Can the remaining values of f be chosen such that f is submodular? This is not a simple question, and recent results suggest that it is quite challenging [53]. Theorem 8 sheds some light on the submodular completion problem, for the special case when each A_i belongs to our particular family \mathcal{A} . Essentially, it says for any b_i 's satisfying $8 \log k \leq b_i \leq |A_i|$, then indeed the remaining values can be chosen such that f is submodular.

Theorem 8 addresses the submodular completion problem via the set family defined in Eq. (1.3), namely

$$\mathcal{I} = \{ I : |I| \leq r \wedge |I \cap A_i| \leq b_i \forall i = 1, \dots, k \}.$$

If \mathcal{I} is the family of independent sets of a matroid, then its rank function might be a good solution to the submodular completion problem. Certainly the rank function will be submodular, and it also satisfies $\text{rank}(A_i) \leq b_i$. But we would like equality to hold. Our question becomes:

$$\text{Is there a matroid satisfying } \text{rank}(A_i) = b_i \text{ for each } i? \tag{4.1}$$

We first discuss two simple cases of this question. Note that \mathcal{I} cannot possibly satisfy (4.1) if $b_i > |A_i|$ or $b_i > r$, so henceforth assume that $b_i \leq \min\{|A_i|, r\}$ for all i .

- **Disjoint A_i 's.** Question (4.1) is quite simple so long as the A_i 's are disjoint. One may check that \mathcal{I} indeed forms a matroid (assuming that $r, b_1, \dots, b_k \geq 0$), and that this matroid satisfies (4.1).
- **Two sets.** Even in the case $k = 2$, question (4.1) is quite interesting. First of all, the family \mathcal{I} above typically does not form a matroid. Consider taking $n = 5$, $r = 4$, $A_1 = \{1, 2, 3\}$, $A_2 = \{3, 4, 5\}$ and $b_1 = b_2 = 2$. Then both $\{1, 2, 4, 5\}$ and $\{2, 3, 4\}$ are maximal sets in \mathcal{I} but they do not have the same cardinality, which violates one of the basic matroid properties.

However, it is a short exercise to check that \mathcal{I} is a matroid if we require that $r \leq b_1 + b_2 - |A_1 \cap A_2|$. (For a proof, see Lemma 28.) This matroid also satisfies (4.1). In fact, we can even relax the constraint $|I| \leq r$ to obtain the family

$$\{ I : |I \cap A_1| \leq b_1 \wedge |I \cap A_2| \leq b_2 \wedge |I \cap (A_1 \cup A_2)| \leq r \},$$

which is also a matroid if $r \leq b_1 + b_2 - |A_1 \cap A_2|$. We would have preferred a somewhat weaker restriction on r , say $r \leq b_1 + b_2$ (which is actually vacuous), but in order to obtain a matroid, this restriction on r must include an ‘‘error term’’ of $-|A_1 \cap A_2|$.

The strategy of Theorem 8 is to generalize this discussion of $k = 2$ to larger values of k . The generalization is not straightforward, as Theorem 8 imposes quite a few conditions on the desired family of matroids. Quite magically, the key to satisfying all of the desired conditions is to ensure that the family \mathcal{A} has strong expansion properties. However there are numerous technical challenges in proving the desired result. Indeed, our first construction, Theorem 9, falls short of the mark as it cannot handle the case $k > n$, whereas proving Theorem 6 requires $k = n^{\omega(1)}$. Theorem 10 improves the first construction with several ideas, and it provides the basis for proving Theorem 8 and Theorem 6.

4.2 The Matroid Constructions

Let $\mathcal{A} = \{A_1, \dots, A_k\}$ be an arbitrary family of sets. Let b_1, \dots, b_k be integers satisfying $0 \leq b_i \leq |A_i|$. As we saw above, in the case $k = 2$,

$$\{ I : |I \cap A_1| \leq b_1 \wedge |I \cap A_2| \leq b_2 \wedge |I \cap (A_1 \cup A_2)| \leq b_1 + b_2 - |A_1 \cap A_2| \} \quad (4.2)$$

is a matroid, where $-|A_1 \cap A_2|$ is an undesirable but necessary “error term” in the last constraint.

To generalize this to $k > 2$, we will impose similar constraints for every subset of the A_i ’s. Our new set family is

$$\mathcal{I} := \left\{ I : \left| I \cap \left(\bigcup_{j \in J} A_j \right) \right| \leq \sum_{j \in J} b_j - \left(\sum_{j \in J} |A_j| - \left| \bigcup_{j \in J} A_j \right| \right) \forall J \subseteq [k] \right\}.$$

In the special case $k = 2$, this is precisely Eq. (4.2). To simplify notation somewhat, let us define the function $f : 2^{[k]} \rightarrow \mathbb{Z}$ where

$$f(J) := \sum_{j \in J} b_j - \left(\sum_{j \in J} |A_j| - |A(J)| \right), \quad \text{and} \quad A(J) := \bigcup_{j \in J} A_j. \quad (4.3)$$

Then we can write more compactly

$$\mathcal{I} = \{ I : |I \cap A(J)| \leq f(J) \forall J \subseteq [k] \}. \quad (4.4)$$

In the definition of $f(J)$, we should again think of $-\left(\sum_{j \in J} |A_j| - |A(J)|\right)$ as an “error term”, since it is non-positive, and it captures the “overlap” of the sets $\{A_j : j \in J\}$. In particular, if $J = \{1, 2\}$ then this error term is precisely $-|A_1 \cap A_2|$, as it was in the case $k = 2$. Furthermore, if the A_j ’s are all disjoint then the error terms are all 0, and so the family \mathcal{I} reduces to $\{ I : |I \cap A_j| \leq b_j \forall j \in [k] \}$, which is simply a partition matroid.

Our first matroid construction is given by the following theorem, which is proven in Appendix E.1.

Theorem 9. *The family \mathcal{I} given in Eq. (4.4) is the family of independent sets of a matroid, if it is non-empty.*

As mentioned above, Theorem 9 does not suffice to prove Theorem 6. To see why, suppose that $k > n$ and that $b_i < |A_i|$ for every i . Then $f([k]) \leq n - k < 0$, and therefore \mathcal{I} is empty. So the construction of Theorem 9 is only applicable when $k \leq n$, which is insufficient for proving Theorem 6.

We now modify the preceding construction by introducing a sort of “truncation” operation which allows us to take $k \gg n$. We emphasize that this truncation is *not* ordinary matroid truncation. The ordinary truncation operation *decreases* the rank of the matroid, whereas we want to *increase* the rank by throwing away constraints in the definition of \mathcal{I} . In particular, we will introduce an additional parameter τ , and only keep constraints for $|J| < \tau$. It turns out that, so long as f is large enough for a certain interval, then we can truncate f and still get a matroid.

Definition 2. *Let μ and τ be non-negative integers. A function $f : 2^{[k]} \rightarrow \mathbb{R}$ is called (μ, τ) -large if*

$$f(J) \geq \begin{cases} 0 & \forall J \subseteq [k], |J| < \tau \\ \mu & \forall J \subseteq [k], \tau \leq |J| \leq 2\tau - 2. \end{cases}$$

The truncated function $\bar{f} : 2^{[k]} \rightarrow \mathbb{Z}$ is defined by

$$\bar{f}(J) := \begin{cases} f(J) & (\text{if } |J| < \tau) \\ \mu & (\text{otherwise}). \end{cases}$$

We now argue that we can truncate f and still obtain a matroid. Our second matroid construction is:

Theorem 10. *Suppose that the function f defined in Eq. (4.3) is (μ, τ) -large. Then the family*

$$\bar{\mathcal{I}} = \{ I : |I \cap A(J)| \leq \bar{f}(J) \ \forall J \subseteq [k] \}$$

is the family of independent sets of a matroid.

If we assume that the A_i 's cover the ground set, i.e., $A([k]) = [n]$, or if we apply ordinary matroid truncation to reduce the rank to μ , then the family $\bar{\mathcal{I}}$ can be written

$$\bar{\mathcal{I}} = \left\{ I : |I| \leq \mu \ \wedge \ |I \cap A(J)| \leq f(J) \ \forall J \subseteq [k], |J| < \tau \right\}.$$

This construction yields quite a broad family of matroids. We list several interesting special cases in Appendix F. In particular, partition matroids and paving matroids are both special cases. Thus, our construction can produce non-linear matroids, as the Vámos matroid is both non-linear and paving.

4.3 Theorem 8 and Matroids from Lossless Expanders

To prove Theorem 8, we must construct the desired set family \mathcal{A} and the matroid family \mathcal{M} . To achieve the desired properties of \mathcal{M} , we require that \mathcal{A} satisfies a strong expansion property which we describe now.

Definition 3. *Let $G = (U \cup V, E)$ be a bipartite graph. For $J \subseteq U$, define*

$$\Gamma(J) := \{ v : \exists u \in U \text{ such that } \{u, v\} \in E \}.$$

For simplicity we let $\Gamma(u) = \Gamma(\{u\})$. The graph G is called a (D, L, ϵ) -lossless expander if

$$\begin{aligned} |\Gamma(u)| &= D \quad \forall u \in U \\ |\Gamma(J)| &\geq (1 - \epsilon) \cdot D \cdot |J| \quad \forall J \subseteq U, |J| \leq L. \end{aligned}$$

Lossless expanders are well-studied [28, 24], and their existence is discussed below in Theorem 14. Given such a G , we will construct our set family $\mathcal{A} = \{A_1, \dots, A_k\} \subseteq 2^{[n]}$ by identifying $U = [k]$, $V = [n]$, and for each vertex $i \in U$ we define A_i to be $\Gamma(i)$. The various parameters in Theorem 8 must be reflected in the various parameters of G , so let us now make clear the relationships between these parameters.

$$U = [k], \quad V = [n], \quad D = \mu, \quad L = 2\tau - 2, \quad \epsilon = \frac{b}{4\mu}, \quad b \geq \frac{2\mu}{\tau}. \quad (4.5)$$

(The actual values are chosen below in Eq. (4.8).) Thus we have:

$$\begin{aligned} |A_i| &= \mu \quad \forall i \in U \\ |A(J)| &\geq (1 - \epsilon) \cdot \mu \cdot |J| \quad \forall J \subseteq U, |J| \leq 2\tau - 2. \end{aligned} \quad (4.6)$$

Recall that we must construct not a single matroid but an entire family of matroids, one for every subfamily $\mathcal{B} \subseteq \mathcal{A}$. Constructing this large number of matroids will be no harder than constructing a single matroid because the matroid properties will follow from the expansion properties of the set family (i.e., Eq. (4.6)), and these properties are obviously preserved by restricting to a subfamily.

For any subfamily $\mathcal{B} \subseteq \mathcal{A}$, we will obtain the matroid $\mathbf{M}_{\mathcal{B}}$ using Theorem 10. Let $U_{\mathcal{B}} \subseteq U$ be the set of indices defining \mathcal{B} , i.e., $\mathcal{B} = \{A_i : i \in U_{\mathcal{B}}\}$. The b_i parameters will all be equal, so let their common value be b . The function defining the matroid is $f_{\mathcal{B}} : 2^{U_{\mathcal{B}}} \rightarrow \mathbb{R}$, defined as in Eq. (4.3) by

$$\begin{aligned} f_{\mathcal{B}}(J) &= \sum_{j \in J} b - \left(\sum_{j \in J} |A_j| - |A(J)| \right) \\ &= (b - \mu)|J| - |A(J)|. \end{aligned}$$

Claim 11. $f_{\mathcal{B}}$ is (μ, τ) -large.

All claims in this section are proven in Appendix E.3. By this claim, Theorem 10 implies that

$$\mathcal{I}_{\mathcal{B}} = \left\{ I : |I| \leq \mu \wedge |I \cap A(J)| \leq f_{\mathcal{B}}(J) \forall J \subseteq U_{\mathcal{B}}, |J| < \tau \right\} \quad (4.7)$$

is the family of independent sets of a matroid, which we call $\mathbf{M}_{\mathcal{B}}$.

The next step in proving Theorem 8 is to analyze $\text{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i)$ for $A_i \in \mathcal{A}$. This is accomplished by the following two claims, which follow from Eq. (4.6) without too much difficulty.

Claim 12. For all $\mathcal{B} \subseteq \mathcal{A}$ and all $A_i \in \mathcal{B}$ we have $\text{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i) = b$, assuming that $b \leq \mu$.

Claim 13. For all $\mathcal{B} \subseteq \mathcal{A}$ and all $A_i \in \mathcal{A} \setminus \mathcal{B}$ we have $\text{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i) = \mu$.

Lastly, we show that the existence of an expander graph G with parameters that are sufficient to prove Theorem 8. The following probabilistic construction is folklore. We thank Atri Rudra for explaining it to us, and for stating it in these general terms. A less general statement along the same lines can be found in Vadhan's survey [57, Theorem 4.4].

Theorem 14. Let $k \geq 2$ and $\epsilon \geq 0$. For any $L \leq k$, let $D \geq 2 \log(k)/\epsilon$ and $n \geq 6LD/\epsilon$. Then a (D, L, ϵ) -lossless expander exists.

To prove Theorem 8, we choose $k = 2^{o(n^{1/3})}$ and $\mu = |A_i| = n^{1/3}$ (cf. Eq. (4.6)). It is clear from Claim 12 that we must choose $b = 8 \log k$. (We require that k is at most $2^{o(n^{1/3})}$ because Claim 12 assumes $b \leq \mu$.) Following Eq. (4.5), we can therefore take

$$D = \mu = n^{1/3}, \quad \epsilon = \frac{b}{4\mu} = \frac{2 \log k}{n^{1/3}}, \quad \tau = \frac{2\mu}{b} = \frac{n^{1/3}}{4 \log k}, \quad L = 2\tau - 2 \leq \frac{n^{1/3}}{2 \log k}. \quad (4.8)$$

These parameters satisfy the hypotheses of Theorem 14, so our desired expander exists and Theorem 8 is proven.

5 An $O(\sqrt{n})$ -approximation Algorithm

In this section we discuss our most general upper bounds for efficiently PMAC-learning two very broad families of functions: a PMAC-learning algorithm with approximation factor $O(n)$ for learning the family of non-negative, monotone, subadditive functions and a PMAC-learning algorithm with approximation factor $O(\sqrt{n})$ for learning the class of non-negative, monotone, submodular functions.

Theorem 15. Let \mathcal{F} be the class of non-negative, monotone, subadditive functions over $X = 2^{[n]}$. There is an algorithm that PMAC-learns \mathcal{F} with approximation factor $n + 1$. That is, for any distribution D over X , for any ϵ, δ sufficiently small, with probability $1 - \delta$, the algorithm produces a function f that approximates f^* within a multiplicative factor of $n + 1$ on a set of measure $1 - \epsilon$ with respect to D . The algorithm uses $\ell = \frac{48n}{\epsilon} \log\left(\frac{9n}{\delta\epsilon}\right)$ training examples and runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$.

Theorem 16. Let \mathcal{F} be the class of non-negative, monotone, submodular functions over $X = 2^{[n]}$. There is an algorithm that PMAC-learns \mathcal{F} with approximation factor $\sqrt{n + 1}$. That is, for any distribution D over X , for any ϵ, δ sufficiently small, with probability $1 - \delta$, the algorithm produces a function f that approximates f^* within a multiplicative factor of $\sqrt{n + 1}$ on a set of measure $1 - \epsilon$ with respect to D . The algorithm uses $\ell = \frac{48n}{\epsilon} \log\left(\frac{9n}{\delta\epsilon}\right)$ training examples and runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$.

The proofs of these two theorems are very similar. To give a feel for our technique, we now sketch a proof of Theorem 15 with a number of simplifying assumptions. We then mention how to modify the proof to obtain Theorem 16. Full proofs of these theorems are given in Appendix G.

Proof Sketch (of Theorem 15). To avoid technicalities, let us assume that $f^*(S) > 0$ whenever $S \neq \emptyset$.

First, using the fact that f^* is subadditive we argue that there exists a linear function $\hat{f}(S) = w^T \chi(S)$ such that $f^*(S) \leq \hat{f}(S) \leq n \cdot f^*(S)$ for all $S \subseteq [n]$. This is equivalent to saying that the following examples

in \mathbb{R}^{n+1} are linearly separable.

$$\begin{aligned} \text{Examples labeled } +1: & \quad \text{ex}_S^+ := (\chi(S), f^*(S)) & \quad \forall S \subseteq [n] \\ \text{Examples labeled } -1: & \quad \text{ex}_S^- := (\chi(S), n \cdot f^*(S)) & \quad \forall S \subseteq [n] \end{aligned}$$

This suggests trying to reduce our learning problem to the standard problem of learning a linear separator for these examples in the standard PAC model (i.e., learning from i.i.d. samples in the passive, supervised learning setting [37, 59]). This is exactly our proof strategy. However, in order to apply standard techniques to learn such a linear separator, we must ensure that our training examples are i.i.d. To achieve this, we create a i.i.d. distribution D' in \mathbb{R}^{n+1} that is related to the original distribution D as follows. First, we draw a sample $S \subseteq [n]$ from the distribution D and then flip a fair coin. The sample from D' is ex_S^+ if the coin is heads and ex_S^- if the coin is tails. Of course, each example ex_S^+ is labeled $+1$ and ex_S^- is labeled -1 . As mentioned above, these labeled examples are linearly separable in \mathbb{R}^{n+1} .

Conversely, suppose we can find a linear separator that classifies most of the examples coming from D' correctly. Assume that this linear separator in \mathbb{R}^{n+1} is defined by the function $u^\top x = 0$, where $u = (w, -z)$, $w \in \mathbb{R}^n$ and $z \in \mathbb{R}$. The key observation is that the function $f(S) = \frac{1}{(n+1)z} w^\top \chi(S)$ approximates f^* to within a factor $n+1$ on most of the points coming from D .

Given these observations, our algorithm is as follows.

1. Using the training set $\mathcal{S} = \{(S_1, f^*(S_1)), \dots, (S_\ell, f^*(S_\ell))\}$ of examples drawn from D , construct a new training sequence $\mathcal{S}' = ((x_1, y_1), \dots, (x_m, y_m))$ of training examples drawn from D' which will be used for learning the linear separator. To do this, pick each $y_i \in \{+1, -1\}$ uniformly at random. If $y_i = +1$ set $x_i = \text{ex}_S^+$ and if $y_i = -1$ set $x_i = \text{ex}_S^-$.
2. Solve a linear program in order to find a linear separator $u^\top x = 0$ consistent with the labeled examples in \mathcal{S}' . Let $w \in \mathbb{R}^n$, $z \in \mathbb{R}$ satisfy $u = (w, z)$.
3. The output hypothesis is $f(S) = \frac{1}{(n+1)z} w^\top \chi(S)$.

To prove correctness, note that the LP is feasible; this follows from our earlier discussion that the labeled examples are linearly separable. To prove that this algorithm PMAC-learns f^* , it suffices to prove that the linear separator obtained from the linear program correctly classifies most of the examples coming from D' . This follows from standard VC-dimension bounds [37, 59]. ■

Proof Sketch (of Theorem 16). The proof for Theorem 16 is almost identical with the proof of Theorem 15, replacing the structural result for subadditive functions with a stronger structural result for submodular functions, which shows that, for any $f \in \mathcal{F}$, the function f^2 can be approximated to within a factor of n by a linear function. The proof of Theorem 15 can then be applied to the family $\{f^2 : f \in \mathcal{F}\}$. ■

Remark. We remark that our algorithm which proves Theorem 16 is significantly simpler than the algorithm of Goemans et al. [21] which achieves a slightly worse approximation factor in the exact learning model with value queries.

Extensions. Our algorithms for learning subadditive and submodular functions in the PMAC model are quite robust and can be extended to handle more general scenarios, including various forms of noise. First, we can extend the results in Theorem 15 and Theorem 16 to the more general case where do not even assume that the target function is subadditive (or submodular), but that it is within a factor α of a subadditive (or submodular) function on every point in the instance space. Under this relaxed assumption we are able to achieve the approximation factor $\alpha \cdot (n+1)$ (or $\sqrt{\alpha \cdot (n+1)}$). We can also extend the results in Theorem 15 and Theorem 16 to the *agnostic case* where we assume that there exists a subadditive (or a submodular) function that agrees with the target on all but an η fraction of the points; note that on the η fraction of the points the target can be arbitrarily far from a subadditive (or a submodular) function. In this case we can still

PMAC-learn with a polynomial number of samples $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta\epsilon}))$, but using a potentially computationally inefficient procedure. For formal theorems and proofs see Appendix G.1.

Finally, it is clear from the proofs of Theorem 15 and Theorem 16 that any improvements in the approximation factor for approximating subadditive or submodular functions by linear functions (i.e., Lemma 30 or Lemma 31) for specific subclasses of subadditive (or submodular) functions yield PMAC-learning algorithms with improved approximation factors.

6 Conclusions

In this paper we study submodular function learning in the traditional distributional learning setting. We prove polynomial upper and lower bounds on the approximability guarantees achievable in the general case by using only a polynomial number of examples drawn i.i.d from the underlying distribution. We also provide improved analyses for important subclasses of submodular functions under natural distributional assumptions.

Our work combines central issues in optimization (submodular functions and matroids) with central issues in learning (learnability of natural but complex classes of functions in a distributional setting). Our analysis brings a twist on the usual learning theory models and uncovers some interesting structural and extremal properties of submodular functions, which are likely to be useful in other contexts as well.

6.1 Open Questions

- It would be interesting to close the gap between the $O(n^{1/2})$ upper bound in Theorem 16 and the $\tilde{\Omega}(n^{1/3})$ lower bound in Theorem 6. We suspect that the lower bound can be improved to $\tilde{\Omega}(n^{1/2})$. If such an improved lower bound is possible, the matroids or submodular functions used in its proof are likely to be very interesting. It would be also be interesting to use the approach in our general \sqrt{n} -upper bound for simplifying the analysis in the upper bound of Goemans et al. [21].
- The algorithm in Section 3 applies to matroid rank functions. It trivially extends to L -Lipschitz functions for any constant L . What if $L \geq n$?
- A result similar to Theorem 2 can be proven even if f is non-monotone, using self-bounding functions. Can our proof of Theorem 2 be generalized to obtain such a result?
- Are there particular subclasses of submodular functions for which one can PMAC-learn with approximation ratio better than $O(\sqrt{n})$, perhaps under additional distributional assumptions? Can one PMAC-learn other natural classes of real-valued functions, with good approximation ratios?

Acknowledgements

We would like to thank Avrim Blum, Jan Vondrák, and Atri Rudra for insightful and stimulating discussions. We also thank Steve Hanneke, Lap Chi Lau, Atri Rudra, Alex Samorodnitsky, Mohit Singh, Santosh Vempala, and Van Vu for helpful discussions.

References

- [1] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 2000.
- [2] K. Amano and A. Maruoka. On learning monotone boolean functions under the uniform distribution. *Theoretical Computer Science*, 350(5):3–12, 2006.
- [3] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

- [4] Y. Bakos and E. Brynjolfsson. Bundling information goods: Pricing, profits, and efficiency. *Management Science*, 45(12), 1999.
- [5] M. F. Balcan, A. Blum, and Y. Mansour. Item pricing for revenue maximization. In *Proceedings of the ACM Conference on Electronic Commerce*, 2009.
- [6] E. Baum and K. Lang. Query learning can work poorly when a human oracle is used. In *IEEE International Joint Conference on Neural Networks*, 1993.
- [7] A. Blum, C. Burch, and J. Langford. On learning monotone boolean functions. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, 1998.
- [8] S. Boucheron, G. Lugosi, and P. Massart. On concentration of self-bounding functions. *Electronic Journal of Probability*, 14:1884–1899, 2009.
- [9] N. H. Bshouty and C. Tamon. On the fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747 – 770, 1996.
- [10] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal on Computing*. To appear.
- [11] C. Chekuri and J. Vondrák. Randomized pipage rounding for matroid polytopes and applications, September 2009. arXiv:0909.4348v1.
- [12] D. Dachman-Soled, K. Lee, T. Malkin, R. Servedio, A. Wan, and H. Wee. Optimal cryptographic hardness of learning monotone functions. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, 2008.
- [13] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.
- [14] S. Dobzinski, N. Nisan, and M. Schapira. Truthful Randomized Mechanisms for Combinatorial Auctions. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 644–652, 2006.
- [15] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, pages 69–87. Gordon and Breach, 1970.
- [16] O. Ekin, P. L. Hammer, and U. N. Peled. Horn functions and submodular boolean functions. *Theoretical Computer Science*, 175(2):257–270, 1997.
- [17] U. Feige, V. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, 2007.
- [18] R. Fuerderer, A. Herrmann, and G. Wuebker. *Optimal Bundling: Marketing Strategies for Improving Economic Performance*. Springer, 2010.
- [19] S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2005.
- [20] G. Goel, C. Karande, P. Tripathi, and L. Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science*, 2009.

- [21] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2009.
- [22] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1993.
- [23] V. Guruswami and P. Raghavendra. Hardness of Learning Halfspaces with Noise. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.
- [24] V. Guruswami, C. Umans, and S. P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *Journal of the ACM*, 56(4), 2009.
- [25] M. T. Hajiaghayi, J. H. Kim, T. Leighton, and H. Räcke. Oblivious routing in directed graphs with random demands. In *Proceedings of STOC*, pages 193–201, 2005.
- [26] W. Hanson and R. Kipp Martin. Optimal bundle pricing. *Management Science*, 36(2), 1990.
- [27] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.
- [28] S. Hoory, N. Linial, , and A. Wigderson. Expander graphs and their applications. *Bulletin of the AMS*, 43:439–561, 2006.
- [29] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. *Journal of the ACM*, 48:761–777, 2001.
- [30] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science*, 2009.
- [31] S. Iwata and J. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2009.
- [32] S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. Wiley-Interscience, 2000.
- [33] S. Jegelka and J. Bilmes. Notes on graph cuts with submodular edge weights. In *Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML)*, December 2009.
- [34] F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays, presented to R. Courant on his 60th Birthday, January 8, 1948*, 1948.
- [35] A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6), 2008.
- [36] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.
- [37] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [38] A. Klivans, R. O’Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer and System Sciences*, 68(4), 2004.
- [39] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

- [40] A. Krause and C. Guestrin. Beyond convexity: Submodularity in machine learning, 2008. <http://www.select.cs.cmu.edu/tutorials/icml08submodularity.html>.
- [41] A. Krause and C. Guestrin. Intelligent information gathering and submodular function optimization, 2009. <http://submodularity.org/ijcai09/index.html>.
- [42] B. Lehmann, D. J. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55:270–296, 2006.
- [43] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3), 1993.
- [44] L. Lovász. Submodular functions and convexity. *Mathematical Programming: The State of the Art*, 1983.
- [45] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
- [46] M. Molloy and B. Reed. *Graph Colouring and the Probabilistic Method*. Springer, 2001.
- [47] M. Narasimhan and J. Bilmes. Local search for balanced submodular clusterings. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.
- [48] V. V. Osokin. On learning monotone boolean functions with irrelevant variables. *Discrete Mathematics and Applications*, 20(3):307–320, 2010.
- [49] J. G. Oxley. *Matroid Theory*. Oxford University Press, 1992.
- [50] P. Rusmevichientong, B. Van Roy, and P. W. Glynn. A nonparametric approach to multiproduct pricing. *Operations Research*, 54(1), 2006.
- [51] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80:346–355, 2000.
- [52] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2004.
- [53] C. Seshadhri and J. Vondrák. Is submodularity testable?, August 2010. arXiv:1008.0831.
- [54] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008.
- [55] M. Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l’I.H.É.S.*, 81(1):73–205, December 1995.
- [56] D. M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.
- [57] S. P. Vadhan. Pseudorandomness i. *Foundations and Trends in Theoretical Computer Science*. To Appear. Available at: <http://userweb.cs.utexas.edu/users/diz/395T/09/salil1.pdf>.
- [58] L.G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

- [59] V. N. Vapnik. *Statistical Learning Theory*. Wiley and Sons, 1998.
- [60] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 2008.
- [61] J. Vondrák. A note on concentration of submodular functions, May 2010. arXiv:1005.2791.
- [62] R. B. Wilson. *Nonlinear Pricing*. Oxford University Press, 1997.
- [63] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.

A Standard Facts about Submodular Functions

Theorem 17. *Given a finite universe U , let S_1, S_2, \dots, S_n be subsets of U . Define $f : 2^{[n]} \rightarrow \mathbb{R}_+$ by*

$$f(A) = |\cup_{i \in A} S_i| \quad \text{for } A \subseteq [n].$$

Then f is monotone and submodular. More generally, for any non-negative weight function $w : U \rightarrow \mathbb{R}_+$, the function f defined by

$$f(A) = w(\cup_{i \in A} S_i) \quad \text{for } A \subseteq [n]$$

is monotone and submodular.

Lemma 18. *The minimizers of any submodular function are closed under union and intersection.*

Proof. Assume that J_1 and J_2 are minimizers for f . By submodularity we have

$$f(J_1) + f(J_2) \geq f(J_1 \cap J_2) + f(J_1 \cup J_2).$$

We also have

$$f(J_1 \cap J_2) + f(J_1 \cup J_2) \geq f(J_1) + f(J_2),$$

so $f(J_1) = f(J_2) = f(J_1 \cap J_2) = f(J_1 \cup J_2)$, as desired. ■

B Boolean Submodular Functions

Proposition 19. *The class of monotone, Boolean-valued, submodular functions is efficiently PMAC-learnable with approximation factor 1.*

Proof. Let $f : 2^{[n]} \rightarrow \{0, 1\}$ be an arbitrary monotone, submodular function. We claim that f is either constant or a monotone disjunction. If $f(\emptyset) = 1$ then this is trivial, so assume $f(\emptyset) = 0$.

As stated in Eq. (1.2), submodularity is equivalent to the property of decreasing marginal values. In particular, since $f(\emptyset) = 0$, we get

$$f(T \cup \{x\}) - f(T) \leq f(\{x\}) \quad \forall T \subseteq [n], x \in [n] \setminus T.$$

If $f(\{x\}) = 0$ then this together with monotonicity implies that $f(T \cup \{x\}) = f(T)$ for all T and all x . On the other hand, if $f(\{x\}) = 1$ then monotonicity implies that $f(T) = 1$ for all T such that $x \in T$. Thus we have argued that f is a disjunction:

$$f(S) = \begin{cases} 1 & \text{(if } S \cap X \neq \emptyset) \\ 0 & \text{(otherwise)} \end{cases},$$

where $X = \{x : f(\{x\}) = 1\}$. This proves the claim.

It is well known that the class of disjunctions is easy to learn in the supervised learning setting [37, 59]. ■

Non-monotone, Boolean, submodular functions need not be disjunctions. For example, consider the function f where $f(S) = 0$ if $S \in \{\emptyset, [n]\}$ and $f(S) = 1$ otherwise; it is submodular, but not a disjunction. However, it turns out that any submodular boolean functions is a 2-DNF. This was already known [16], and it can be proven by case analysis as in Proposition 19. It is well known that 2-DNFs are efficiently PAC-learnable. We summarize this discussion as follows.

Proposition 20. *The class of Boolean-valued, submodular functions is efficiently PMAC-learnable with approximation factor 1.*

C Lipschitz Functions and Product Distributions

In this section we show that Lipschitz functions can be PMAC-learned when the distribution on examples is a product distribution.

C.1 Proof of Theorem 1

Theorem 1. *Let \mathcal{F} be the class of non-negative, monotone, 1-Lipschitz, submodular functions with ground set $[n]$ and minimum non-zero value η . Let D be a product distribution on $[n]$. For any sufficiently small $\epsilon > 0$ and $\delta > 0$, Algorithm 1 PMAC-learns \mathcal{F} with approximation factor $O(\log(1/\epsilon)/\eta)$. The number of training examples used is $10n^2 \log(1/\delta) + n \log(n/\delta)/\epsilon$.*

Proof. To formally analyze this algorithm, let us first consider the estimate μ . Note that $0 \leq f(X) \leq n$ for all X . So a Hoeffding bound implies that, with probability at least $1 - \delta$,

$$\begin{aligned} \mu \geq 450 \log(1/\epsilon) &\implies \mathbf{E}[f(X)] \geq 400 \log(1/\epsilon) \quad \text{and} \quad \frac{5}{6} \mathbf{E}[f(X)] \leq \mu \leq \frac{4}{3} \mathbf{E}[f(X)] \\ \mu < 450 \log(1/\epsilon) &\implies \mathbf{E}[f(X)] \leq 500 \log(1/\epsilon). \end{aligned}$$

Case 1: Assume that $\mathbf{E}[f(X)] \geq 400 \log(1/\epsilon)$; this holds with probability at least $1 - \delta$. If ϵ is sufficiently small then Corollary 3 implies

$$\begin{aligned} \Pr[\mu/4 \leq f(X) \leq 2\mu] &\geq \Pr\left[\frac{1}{3} \mathbf{E}[f(X)] \leq f(X) \leq \frac{5}{3} \mathbf{E}[f(X)]\right] \\ &\geq 1 - \Pr[|f(X) - \mathbf{E}[f(X)]| \geq (2/3) \mathbf{E}[f(X)]] \quad (\text{C.1}) \\ &\geq 1 - 4e^{-\mathbf{E}[f(X)]/100} \geq 1 - \epsilon. \end{aligned}$$

Therefore, with confidence at least $1 - \delta$, the algorithm achieves approximation factor 8 on all but an ϵ fraction of the distribution.

Case 2: As described earlier, we must separately handle the zeros and the non-zeros. To that end, define

$$\mathcal{P} = \{S : f^*(S) > 0\} \quad \text{and} \quad \mathcal{Z} = \{S : f^*(S) = 0\}.$$

Recall that $U = \bigcup_{f^*(S_i)=0} S_i$. Monotonicity and submodularity imply that $f^*(U) = 0$. Letting $\mathcal{L} = \{T : T \subseteq U\}$, monotonicity implies that

$$f^*(T) = 0 \quad \forall T \in \mathcal{L}. \quad (\text{C.2})$$

Let S be a new sample from D and let \mathcal{E} be the event that S violates the inequality

$$f(S) \leq f^*(S) \leq 1200 \log(1/\epsilon) f(S).$$

Our goal is to show that, with probability $1 - \delta$ over the training examples, we have $\Pr[\mathcal{E}] \leq \epsilon$. Clearly

$$\Pr[\mathcal{E}] = \Pr[\mathcal{E} \wedge S \in \mathcal{P}] + \Pr[\mathcal{E} \wedge S \in \mathcal{Z}].$$

We will separately analyze these two probabilities.

First suppose that $S \in \mathcal{P}$, i.e., $f^*(S) \geq \eta$. By Eq. (C.2), $S \not\subseteq U$ and hence our hypothesis has $f(S) = \eta$. Therefore the event $\mathcal{E} \wedge S \in \mathcal{P}$ occurs only when $f^*(S) > 1200 \log(1/\epsilon)$. Assume that $\mathbf{E}[f(X)] \leq 500 \log(1/\epsilon)$; this holds with probability at least $1 - \delta$. We now apply Theorem 2 with

$b = 1200 \log(1/\epsilon)$ and $t = 4\sqrt{\log(1/\epsilon)}$. Then $b - t\sqrt{b} \geq 1000 \log(1/\epsilon) \geq 2 \mathbf{E}[f(X)]$. By Markov's inequality, $\Pr[f(X) \leq b - t\sqrt{b}] \geq 1/2$. So, using Theorem 2, we obtain

$$\Pr[\mathcal{E} \wedge S \in \mathcal{P}] \leq \Pr[f(X) \geq b] \leq 2 \exp(-t^2/4) \leq \epsilon.$$

Next suppose that $S \in \mathcal{Z}$, i.e., $f^*(S) = 0$. Since our hypothesis has $f(S) = 0$ for all $S \in \mathcal{L}$, the event $\mathcal{E} \wedge S \in \mathcal{Z}$ holds only if $S \in \mathcal{Z} \setminus \mathcal{L}$. The proof now follows from Claim 21. \blacksquare

Claim 21. *With probability at least $1 - \delta$, the set $\mathcal{Z} \setminus \mathcal{L}$ has measure at most ϵ .*

Proof. The idea of the proof is as follows. At any stage of the algorithm, we can compute the set U and the subcube \mathcal{L} . We refer to \mathcal{L} as the algorithm's *null subcube*. Suppose that there is at least an ϵ chance that a new example is a zero of f^* , but does not lie in the null subcube. Then such a example should be seen in the next sequence of $\log(1/\delta)/\epsilon$ examples, with probability at least $1 - \delta$. This new example increases the dimension of the null subcube by at least one, and therefore this can happen at most n times.

Formally, for $k \leq \ell$, define

$$U_k = \bigcup_{\substack{i \leq k \\ f^*(S_i)=0}} S_i \quad \text{and} \quad \mathcal{L}_k = \{S : S \subseteq U_k\}.$$

As argued above, we have $\mathcal{L}_k \subseteq \mathcal{Z}$ for any k . Suppose that, for some k , the set $\mathcal{Z} \setminus \mathcal{L}_k$ has measure at least ϵ . Define $k' = k + \log(n/\delta)/\epsilon$. Then amongst the subsequent examples $S_{k+1}, \dots, S_{k'}$, the probability that none of them lie in $\mathcal{Z} \setminus \mathcal{L}_k$ is at most

$$(1 - \epsilon)^{\log(n/\delta)/\epsilon} \leq \delta/n.$$

On the other hand, if one of them does lie in $\mathcal{Z} \setminus \mathcal{L}_k$, then $|\mathcal{U}_{k'}| > |\mathcal{U}_k|$. But $|\mathcal{U}_k| \leq n$ for all k , so this can happen at most n times. Since $\ell \geq n \log(n/\delta)/\epsilon$, with probability at least δ the set $\mathcal{Z} \setminus \mathcal{L}_\ell$ has measure at most ϵ . \blacksquare

C.2 Proof of Theorem 2

Before diving into the proof, we remark that the theorem is much easier to prove in the special case that f is integer-valued. Together with our other hypotheses on f , this implies that f must actually be a matroid rank function. Whenever $f(S)$ is large, this fact can “certified” by any maximal independent subset of S . The theorem then follows easily from a version of Talagrand's inequality which leverages this certification property; see, e.g., [1, §7.7] or [46, §10.1].

We now prove the theorem in its full generality. We may assume that $t \leq \sqrt{b}$, otherwise the theorem is trivial as the left-hand side of Eq. (3.1) is zero. Talagrand's inequality states: for any $\mathcal{A} \subseteq \{0, 1\}^n$ and $y \in \{0, 1\}^n$ drawn from a product distribution,

$$\Pr[y \in \mathcal{A}] \cdot \Pr[\rho(\mathcal{A}, y) > t] \leq \exp(-t^2/4), \tag{C.3}$$

where ρ is a distance function defined by

$$\rho(\mathcal{A}, y) = \sup_{\substack{\alpha \in \mathbb{R}^n \\ \|\alpha\|_2=1}} \min_{z \in \mathcal{A}} \sum_{i: y_i \neq z_i} \alpha_i.$$

We will apply this inequality to the set $\mathcal{A} \subseteq 2^V$ defined by $\mathcal{A} = \{X : f(X) < b - t\sqrt{b}\}$.

Claim 22. *For every $Y \subseteq V$, $f(Y) \geq b$ implies $\rho(\mathcal{A}, Y) > t$.*

Proof. Suppose to the contrary that $\rho(\mathcal{A}, Y) \leq t$. By relabeling, we can write Y as $Y = \{1, \dots, k\}$. For

$i \in \{0, \dots, k\}$, let $E_i = \{1, \dots, i\}$. Define

$$\alpha_i = \begin{cases} f(E_i) - f(E_{i-1}) & (\text{if } i \in Y) \\ 0 & (\text{otherwise}). \end{cases}$$

Since f is monotone and 1-Lipschitz, we have $0 \leq \alpha_i \leq 1$. Thus $\|\alpha\|_2 \leq \sqrt{\sum_i \alpha_i} \leq \sqrt{f(Y)}$, by non-negativity of f .

The definition of ρ and our supposition $\rho(\mathcal{A}, Y) \leq t$ imply that there exists $Z \in \mathcal{A}$ with

$$\sum_{i \in (Y \setminus Z) \cup (Z \setminus Y)} \alpha_i \leq \rho(\mathcal{A}, Y) \cdot \|\alpha\|_2 \leq t\sqrt{f(Y)}. \quad (\text{C.4})$$

We may assume that $Z \subset Y$, since $Z \cap Y$ also satisfies the desired conditions. This follows since monotonicity of f implies that $\alpha \geq 0$ and that \mathcal{A} is downwards-closed.

We will obtain a contradiction by showing that $f(Y) - f(Z) \leq t\sqrt{f(Y)}$. First let us order $Y \setminus Z$ as $(\phi(1), \dots, \phi(m))$, where $\phi(i) < \phi(j)$ iff $i < j$. Next, define $F_i = Z \cup \{\phi(1), \dots, \phi(i)\} \subseteq Y$. Note that $E_j \subseteq F_{\phi^{-1}(j)}$; this follows from our choice of ϕ , since $Z \subseteq F_{\phi^{-1}(j)}$ but we might have $Z \not\subseteq E_j$. Therefore

$$\begin{aligned} f(Y) - f(Z) &= \sum_{i=1}^m (f(F_i) - f(F_{i-1})) \\ &= \sum_{j \in Y \setminus Z} (f(F_{\phi^{-1}(j)}) - f(F_{\phi^{-1}(j)-1})) \\ &\leq \sum_{j \in Y \setminus Z} (f(E_j) - f(E_{j-1})) \quad (\text{since } E_j \subseteq F_{\phi^{-1}(j)} \text{ and } f \text{ is submodular}) \\ &= \sum_{j \in Y \setminus Z} \alpha_j \\ &\leq t\sqrt{f(Y)} \quad (\text{by Eq. (C.4)}). \end{aligned}$$

So $f(Z) \geq f(Y) - t\sqrt{f(Y)} \geq b - t\sqrt{b}$, since $f(Y) \geq b$ and $t \leq \sqrt{b}$. This contradicts $Z \in \mathcal{A}$. \square

This claim implies $\Pr[f(Y) \geq b] \leq \Pr[\rho(\mathcal{A}, Y) > t]$, so the theorem follows from Eq. (C.3).

D An Approximate Characterization of Matroid Rank Functions

Theorem 5. *Let $f : 2^{[n]} \rightarrow \mathbb{Z}_+$ be the rank function of a matroid with no loops, i.e., $f(S) \geq 1$ whenever $S \neq \emptyset$. Fix $\epsilon > 0$, sufficiently small. There exists a concave function $h : [0, n] \rightarrow \mathbb{R}$ such that, for every $k \in [n]$, and for a $1 - \epsilon$ fraction of the sets $S \in \binom{[n]}{k}$, we have*

$$\frac{1}{O(\log(1/\epsilon))} h(k) \leq f(S) \leq O(\log(1/\epsilon)) h(k).$$

Henceforth, we will use the following notation. For $p \in [0, 1]$, let $R(p) \subseteq [n]$ denote the random variable obtained by choosing each element of $[n]$ independently with probability p . For $k \in [n]$, let $S(k) \subseteq [n]$ denote a set of cardinality k chosen uniformly at random. Define the function $h' : [0, 1] \rightarrow \mathbb{R}$ by

$$h'(p) = \mathbf{E}[f(R(p))].$$

For any $\tau \in \mathbb{R}$, define the functions $g_\tau : [0, 1] \rightarrow \mathbb{R}$ and $g'_\tau : [n] \rightarrow \mathbb{R}$ by

$$\begin{aligned} g_\tau(p) &= \Pr[f(R(p)) > \tau] \\ g'_\tau(k) &= \Pr[f(S(k)) > \tau]. \end{aligned}$$

Finally, let us introduce the notation $X \cong Y$ to denote that random variables X and Y are identically distributed.

Lemma 23. h' is concave.

Proof. One way to prove this is by appealing to the *multilinear extension* of f , which has been of great value in recent work [10]. This is the function $F : [0, 1]^{[n]} \rightarrow \mathbb{R}$ defined by $F(y) = \mathbf{E}[f(\hat{y})]$, where $\hat{y} \in \{0, 1\}^{[n]}$ is a random variable obtained by independently setting $\hat{y}_i = 1$ with probability y_i , and $\hat{y}_i = 0$ otherwise. Then $h'(p) = F(p, \dots, p)$. It is known [10] that $\frac{\partial^2 F}{\partial y_i \partial y_j} \leq 0$ for all i, j . By basic calculus, this implies that the second derivative of h' is non-positive, and hence h' is concave. ■

Lemma 24. g'_τ is a monotone function.

Proof. Fix $k \in [n - 1]$ arbitrarily. Pick a set $S = S(k)$. Construct a new set T by adding to S a uniformly chosen element of $V \setminus S$. By monotonicity of f we have $f(S) > \tau \implies f(T) > \tau$. Thus $\Pr[f(S) > \tau] \leq \Pr[f(T) > \tau]$. Since $T \cong S(k+1)$, this implies that $g_\tau(k) \leq g_\tau(k+1)$, as required. ■

Lemma 25. $g'_\tau(k) \leq 2 \cdot g_\tau(k/n)$, for all $\tau \in \mathbb{R}$ and $k \in [n]$.

Proof. This lemma is reminiscent of a well-known property of the Poisson approximation [45, Theorem 5.10], and the proof is also similar. Let $p = k/n$. Then

$$\begin{aligned} g_\tau(p) &= \Pr[f(R(p)) > \tau] \\ &= \sum_{i=0}^n \Pr[f(R(p)) > \tau \mid |R(p)| = i] \cdot \Pr[|R(p)| = i] \\ &= \sum_{i=0}^n g'_\tau(i) \cdot \Pr[|R(p)| = i] \\ &\geq \sum_{i=k}^n g'_\tau(k) \cdot \Pr[|R(p)| = i] \quad (\text{by Lemma 24}) \\ &= g'_\tau(k) \cdot \Pr[|R(p)| \geq k] \\ &\geq g'_\tau(k)/2, \end{aligned}$$

since the mean k of the binomial distribution $B(n, k/n)$ is also a median. ■

Proof (of Theorem 5). For $x \in [0, n]$, define $h(x) = h'(x/n) = \mathbf{E}[f(R(x/n))]$. Fix $k \in [n]$ arbitrarily.

Case 1. Suppose that $h(k) \geq 400 \log(1/\epsilon)$. As argued in Eq. (C.1),

$$\Pr\left[f(R(k/n)) < \frac{1}{3}h(k)\right] \leq \epsilon \quad \text{and} \quad \Pr\left[f(R(k/n)) > \frac{5}{3}h(k)\right] \leq \epsilon.$$

By Lemma 25, $\Pr[f(S(k)) > \frac{5}{3}h(k)] \leq 2\epsilon$. By a symmetric argument, which we omit, one can show that $\Pr[f(S(k)) < \frac{1}{3}h(k)] \leq 2\epsilon$. Thus,

$$\Pr\left[\frac{1}{3}h(k) \leq f(S(k)) \leq \frac{5}{3}h(k)\right] \geq 1 - 4\epsilon.$$

This completes the proof of Case 1.

Case 2. Suppose that $h(k) < 400 \log(1/\epsilon)$. This immediately implies that

$$\Pr\left[f(S(k)) < \frac{h(k)}{400 \log(1/\epsilon)}\right] \leq \Pr[f(S(k)) < 1] = 0, \quad (\text{D.1})$$

since since $k \geq 1$, and since we assume that $f(S) \geq 1$ whenever $S \neq \emptyset$. These same assumptions lead to the following lower bound on h :

$$h(k) \geq \Pr[f(R(k/n)) \geq 1] \geq \Pr[R(k/n) \neq \emptyset] \geq 1 - 1/e. \quad (\text{D.2})$$

Thus

$$\begin{aligned}
& \Pr [f(S(k)) > (2000 \log(1/\epsilon))h(k)] \\
& \leq 2 \cdot \Pr [f(R(k/n)) > (2000 \log(1/\epsilon))h(k)] \quad (\text{by Lemma 25}) \\
& \leq 2 \cdot \Pr [f(R(k/n)) > 1200 \log(1/\epsilon)] \quad (\text{by Eq. (D.2)}) \\
& \leq 2 \cdot \epsilon,
\end{aligned}$$

which can be proven using the concentration result in Corollary 3. Thus,

$$\Pr \left[\frac{h(k)}{400 \log(1/\epsilon)} \leq f(S(k)) \leq (2000 \log(1/\epsilon))h(k) \right] \geq 1 - 2\epsilon,$$

completing the proof of Case 2. \blacksquare

E Inapproximability under Arbitrary Distributions

E.1 Proofs of Theorem 9 and Theorem 10

In this section, we will prove Theorem 9 and Theorem 10. Both of these proofs are based on a useful lemma which describes a very general set of conditions that suffice to obtain a matroid. Surprisingly, it seems that this lemma was not previously known.

Let \mathcal{C} be a family of sets and let $f : \mathcal{C} \rightarrow \mathbb{Z}$ be a function. Consider the family

$$\mathcal{I} = \{ I : |I \cap C| \leq f(C) \ \forall C \in \mathcal{C} \}. \quad (\text{E.1})$$

For any $I \in \mathcal{I}$, define $T(I) = \{ C \in \mathcal{C} : |I \cap C| = f(C) \}$ to be the set of tight constraints. Suppose that f has the following uncrossing property:

$$\forall I \in \mathcal{I}, \ C_1, C_2 \in T(I) \implies (C_1 \cup C_2 \in T(I)) \vee (C_1 \cap C_2 = \emptyset). \quad (\text{E.2})$$

Note that we do not require that $C_1 \cap C_2 \in \mathcal{C}$. Our first observation is that this uncrossing property is sufficient to obtain a matroid.

Lemma 26. *\mathcal{I} is the family of independent sets of a matroid, if it is non-empty.*

Proof. We will show that \mathcal{I} satisfies the required axioms of an independent set family. If $I \subseteq I' \in \mathcal{I}$ then clearly $I \in \mathcal{I}$ also. So suppose that $I \in \mathcal{I}$, $I' \in \mathcal{I}$ and $|I| < |I'|$. Let C_1, \dots, C_m be the maximal sets in $T(I)$ and let $C^* = \cup_i C_i$. Note that these maximal sets are disjoint, otherwise we could replace them with their union. In other words, $C_i \cap C_j = \emptyset$ for $i \neq j$, otherwise Eq. (E.2) implies that $C_i \cup C_j \in T(I)$, contradicting maximality. So

$$|I' \cap C^*| = \sum_{i=1}^m |I' \cap C_i| \leq \sum_{i=1}^m f(C_i) = \sum_{i=1}^m |I \cap C_i| = |I \cap C^*|.$$

Since $|I'| > |I|$ but $|I' \cap C^*| \leq |I \cap C^*|$, we must have that $|I' \setminus C^*| > |I \setminus C^*|$. The key consequence is that some element $x \in I' \setminus I$ is not contained in any tight set, i.e., there exists $x \in I' \setminus (C^* \cup I)$. Then $I + x \in \mathcal{I}$ because for every $C \ni x$ we have $|I \cap C| \leq f(C) - 1$. \blacksquare

We remark that this lemma implies an alternative proof of a result of Edmonds, which we do not use in this paper. Corollary 27 does not seem to be sufficient to prove Theorem 8.

Corollary 27 (Edmonds [15], Theorem 15). *Let \mathcal{C} be an intersecting family and $f : \mathcal{C} \rightarrow \mathbb{R}$ an intersecting-submodular function. Then \mathcal{I} as defined in Eq. (E.1) is the family of independent sets of a matroid, if it is non-empty.*

We now use Lemma 26 to prove Theorem 9.

Theorem 9. *The family \mathcal{I} defined in Eq. (4.4), namely*

$$\mathcal{I} = \{ I : |I \cap A(J)| \leq f(J) \ \forall J \subseteq [k] \},$$

where

$$f(J) := \sum_{j \in J} b_j - \left(\sum_{j \in J} |A_j| - |A(J)| \right) \quad \text{and} \quad A(J) := \bigcup_{j \in J} A_j,$$

is the family of independent sets of a matroid, if it is non-empty.

This theorem is proven by uncrossing the constraints defining \mathcal{I} and applying Lemma 26. It is not a priori obvious that the constraints can be uncrossed because both the left-hand side $|I \cap A(J)|$ and the right-hand side $f(J)$ are submodular functions of J . In typical uses of uncrossing, the left-hand side is *supermodular*. The following proof is a simplification of our original proof, due to Jan Vondrák.

Proof. We will apply Lemma 26 to the family $\mathcal{C} = \{ A(J) : J \subseteq [k] \}$ and the function $f' : \mathcal{C} \rightarrow \mathbb{Z}$ defined by

$$f'(C) := \min \{ f(J) : A(J) = C \}.$$

Fix $I \in \mathcal{I}$ and suppose that C_1 and C_2 are tight, i.e., $|I \cap C_i| = f'(C_i)$. For $i \in \{1, 2\}$, let J_i satisfy $C_i = A(J_i)$ and $f'(C_i) = f(J_i)$. Define $h_I : 2^{[k]} \rightarrow \mathbb{Z}$ by

$$h_I(J) := f(J) - |I \cap A(J)| = |A(J) \setminus I| - \sum_{j \in J} (|A_j| - b_j).$$

We claim that h_I is a submodular function of J . This follows because $J \mapsto |A(J) \setminus I|$ is a submodular function of J (cf. Theorem 17 in Appendix A), and $J \mapsto \sum_{j \in J} (|A_j| - b_j)$ is a modular function of J .

Since $I \in \mathcal{I}$ we have $|I \cap A(J)| \leq f(J)$, implying $h_I \geq 0$. But, for $i \in \{1, 2\}$,

$$h_I(J_i) = f(J_i) - |I \cap A(J_i)| = f'(C_i) - |I \cap C_i| = 0,$$

so J_1 and J_2 are minimizers of h_I . It is well-known that the minimizers of any submodular function are closed under union and intersection. (See Lemma 18 in Appendix A.) So $J_1 \cup J_2$ and $J_1 \cap J_2$ are also minimizers, implying that $A(J_1 \cup J_2) = A(J_1) \cup A(J_2) = C_1 \cup C_2$ is also tight.

This shows that Eq. (E.2) holds, so the theorem follows from Lemma 26. ■

A similar approach is used for our second construction.

Theorem 10. *Suppose that the function f defined in Eq. (4.3) is (μ, τ) -large. Then the family*

$$\bar{\mathcal{I}} = \{ I : |I \cap A(J)| \leq \bar{f}(J) \quad \forall J \subseteq [k] \}$$

is the family of independent sets of a matroid.

Proof. Fix $I \in \bar{\mathcal{I}}$. Let J_1 and J_2 satisfy $|I \cap A(J_i)| = \bar{f}(J_i)$. By considering two cases, we will show that $|I \cap A(J_1 \cup J_2)| \geq \bar{f}(J_1 \cup J_2)$, so the desired result follows from Lemma 26.

Case 1: $\max \{|J_1|, |J_2|\} \geq \tau$. Without loss of generality, $|J_1| \geq |J_2|$. Then

$$\bar{f}(J_1 \cup J_2) = \mu = \bar{f}(J_1) = |I \cap A(J_1)| \leq |I \cap A(J_1 \cup J_2)|.$$

Case 2: $\max \{|J_1|, |J_2|\} \leq \tau - 1$. So $|J_1 \cup J_2| \leq 2\tau - 2$. We have $|I \cap A(J_i)| = \bar{f}(J_i) = f(J_i)$ for both i . As argued in Theorem 9, we also have $|I \cap A(J_1 \cup J_2)| = f(J_1 \cup J_2)$. But $f(J_1 \cup J_2) \geq \bar{f}(J_1 \cup J_2)$ since f is (μ, τ) -large. ■

E.2 Proofs of Theorem 6 and Corollary 7

Theorem 6. *No algorithm can PMAC-learn the class of non-negative, monotone, submodular functions with approximation factor $o(n^{1/3}/\log n)$.*

Proof. We will apply Theorem 8 with $k = 2^d$ where $d = \omega(\log n)$. Let \mathcal{A} and \mathcal{M} be the families constructed by Theorem 8. Let the underlying distribution D on $2^{[n]}$ be the uniform distribution on \mathcal{A} . (Note that D is not a product distribution.) Choose a matroid $\mathbf{M}_{\mathcal{B}} \in \mathcal{M}$ uniformly at random and let the target function be

$f^* = \text{rank}_{\mathcal{M}_{\mathcal{B}}}$. Consider any algorithm which attempts to PMAC-learn f^* ; note that the algorithm does not know \mathcal{B} . For any $A \in \mathcal{A}$ that is not a training example, the algorithm has *no information* about $f^*(A)$, so it cannot determine its value better than randomly guessing between the two possible values $16d$ and $|A|$. The set of non-training examples has measure $1 - 2^{-d+O(\log n)}$. So the expected measure of the set on which the algorithm correctly determines the rank is at most $1/2 + 2^{-d+O(\log n)}$. On the set for which the algorithm did not correctly determine the rank, its approximation factor can be no better than $n^{1/3}/(16d)$. ■

Corollary 7. *Suppose that one-way functions exist. For any constant $\epsilon > 0$, no algorithm can PMAC-learn the class of non-negative, monotone, submodular functions with approximation factor $O(n^{1/3-\epsilon})$, even if the functions are given via polynomial-time algorithms which compute their value on the support of the distribution.*

Proof. The argument follows Kearns-Valiant [36]. We will apply Theorem 8 with $k = 2^d$ where $d = n^\epsilon$. There exists a family of pseudorandom Boolean functions $F_d = \{f_y : y \in \{0, 1\}^d\}$, where each function is of the form $f_y : \{0, 1\}^d \rightarrow \{0, 1\}$. Choose an arbitrary bijection between $\{0, 1\}^d$ and \mathcal{A} . Then each $f_y \in F_d$ corresponds to some subfamily $\mathcal{B} \subseteq \mathcal{A}$, and hence to a matroid rank function $\text{rank}_{\mathcal{M}_{\mathcal{B}}}$. Suppose there is a PMAC-learning algorithm for this family of functions which achieves approximation ratio better than $n^{1/3}/16d$ on a set of measure $1/2 + 1/\text{poly}(n)$. Then this algorithm must be predicting the function f_y on a set of size $1/2 + 1/\text{poly}(n) = 1/2 + 1/\text{poly}(d)$. This is impossible, since the family F_d is pseudorandom. ■

E.3 Additional Proofs for Theorem 8

Claim 11. $f_{\mathcal{B}}$ is (μ, τ) -large.

Proof. Consider $J \subseteq U_{\mathcal{B}}$, $\tau \leq |J| \leq 2\tau - 2$. Then

$$\begin{aligned} f_{\mathcal{B}}(J) &= (b - \mu)|J| + |A(J)| \\ &\geq b|J| - \epsilon\mu|J| \quad (\text{by Eq. (4.6) and } |J| \leq 2\tau - 2) \\ &= \frac{3b}{4}|J| \quad (\text{since } \epsilon = b/4\mu) \\ &\geq \mu \quad (\text{since } b \geq 2\mu/\tau \text{ and } |J| \geq \tau). \end{aligned} \tag{E.3}$$

So, $f_{\mathcal{B}}$ is (μ, τ) -large. ■

Claim 12. *For all $\mathcal{B} \subseteq \mathcal{A}$ and all $A_i \in \mathcal{B}$ we have $\text{rank}_{\mathcal{M}_{\mathcal{B}}}(A_i) = b$, assuming that $b \leq \mu$.*

Proof. The definition of $\mathcal{I}_{\mathcal{B}}$ includes the constraint $|I \cap A_i| \leq f_{\mathcal{B}}(\{i\}) = b$. This immediately implies $\text{rank}_{\mathcal{M}_{\mathcal{B}}}(A_i) \leq b$. To prove that equality holds, it suffices to prove that $f_{\mathcal{B}}(J) \geq b$ whenever $|J| \geq 1$, since this implies that every constraint in the definition of $\mathcal{I}_{\mathcal{B}}$ has right-hand side at least b (except when $J = \emptyset$, and assuming that $\mu \geq b$). For $|J| = 1$ this is immediate, and for $|J| \geq 2$ we have

$$f_{\mathcal{B}}(J) = b|J| - \mu|J| + |A(J)| \geq b|J| - \epsilon\mu|J| = b|J| - b|J|/4 \geq b,$$

which completes the proof. ■

Claim 13. *For all $\mathcal{B} \subseteq \mathcal{A}$ and all $A_i \in \mathcal{A} \setminus \mathcal{B}$ we have $\text{rank}_{\mathcal{M}_{\mathcal{B}}}(A_i) = \mu$.*

Proof. Since $\mu = |A_i|$, the condition $\text{rank}_{\mathcal{M}_{\mathcal{B}}}(A_i) = \mu$ holds iff $A_i \in \mathcal{I}_{\mathcal{B}}$. So it suffices to prove that A_i satisfies all constraints in the definition of $\mathcal{I}_{\mathcal{B}}$ (in Eq. (4.7)).

The constraint $|A_i| \leq \mu$ is trivially satisfied, by Eq. (4.6). So it remains to show that for every $J \subseteq U_{\mathcal{B}}$ with $|J| < \tau$, we have

$$|A_i \cap A(J)| \leq f_{\mathcal{B}}(J). \quad (\text{E.4})$$

This is trivial if $J = \emptyset$, so assume $|J| \geq 1$. We have

$$\begin{aligned} |A_i \cap A(J)| &= |A_i| + |A(J)| - |A(J+i)| \\ &\leq \mu + \mu|J| - (1 - \epsilon)\mu|J+i| \quad (\text{by Eq. (4.6)}) \\ &= \frac{b|J+i|}{4} \\ &\leq \frac{b|J|}{2} \\ &\leq f_{\mathcal{B}}(J) \quad (\text{by Eq. (E.3)}). \end{aligned}$$

This proves Eq. (E.4), so $A_i \in \mathcal{I}_{\mathcal{B}}$, as desired. ■

F Special Cases of the Matroid Construction

The matroid constructions of Theorem 9 and Theorem 10 have several interesting special cases.

F.1 Partition Matroids

We are given disjoint sets A_1, \dots, A_k and values b_1, \dots, b_k . We claim that the matroid \mathcal{I} defined in Theorem 9 is a partition matroid. To see this, note that $f(J) = \sum_{j \in J} b_j$, since the A_j 's are disjoint, so f is a modular function. Similarly, $|I \cap A(J)|$ is a modular function of J . Thus, whenever $|J| > 1$, the constraint $|I \cap A(J)| \leq f(J)$ is redundant — it is implied by the constraints $|I \cap A_j| \leq b_j$ for $j \in J$. So we have

$$\mathcal{I} = \{ I : |I \cap A(J)| \leq f(J) \ \forall J \subseteq [k] \} = \{ I : |I \cap A_j| \leq b_j \ \forall j \in [k] \},$$

which is the desired partition matroid.

F.2 Pairwise Intersections

We are given sets A_1, \dots, A_k and values b_1, \dots, b_k . We now describe the special case of the matroid construction which only considers the pairwise intersections of the A_i 's.

Lemma 28. *Let μ be a non-negative integer such that $\mu \leq \min_{i,j \in [k]} (b_i + b_j - |A_i \cap A_j|)$. Then*

$$\mathcal{I} = \{ I : |I| \leq \mu \wedge |I \cap A_j| \leq b_j \ \forall j \in [k] \}$$

is the family of independent sets of a matroid.

Proof. Note that for any pair $J = \{i, j\}$, we have $f(J) = b_i + b_j - |A_i \cap A_j|$. Then

$$\mu \leq \min_{i,j \in [k]} (b_i + b_j - |A_i \cap A_j|) = \min_{J \subseteq [k], |J|=2} f(J),$$

so f is $(\mu, 2)$ -large. The lemma follows from Theorem 10. ■

F.3 Paving Matroids

A *paving matroid* is defined to be a matroid $\mathbf{M} = (V, \mathcal{I})$ of rank m such that every circuit has cardinality either m or $m + 1$. We will show that every paving matroid can be derived from our matroid construction (Theorem 10). First of all, we require a structural lemma about paving matroids.

Lemma 29. *Let $\mathbf{M} = (V, \mathcal{I})$ be a paving matroid of rank m . There exists a family $\mathcal{A} = \{A_1, \dots, A_k\} \subset 2^V$ such that*

$$\mathcal{I} = \{ I : |I| \leq m \wedge |I \cap A_i| \leq m - 1 \ \forall i \} \quad (\text{F.1a})$$

$$|A_i \cap A_j| \leq m - 2 \ \forall i \neq j \quad (\text{F.1b})$$

Proof. It is easy to see that there exists \mathcal{A} satisfying Eq. (F.1a), since we may simply take \mathcal{A} to be the family of circuits which have size m . So let us choose a family \mathcal{A} that satisfies Eq. (F.1a) and minimizes $|\mathcal{A}|$. We will show that this family must satisfy Eq. (F.1b). Suppose otherwise, i.e., there exist $i \neq j$ such that $|A_i \cap A_j| \geq m - 1$.

Case 1: $r(A_i \cup A_j) \leq m - 1$. Then $\mathcal{A} \setminus \{A_i, A_j\} \cup \{A_i \cup A_j\}$ also satisfies Eq. (F.1a), contradicting minimality of $|\mathcal{A}|$.

Case 2: $r(A_i \cup A_j) = m$. Observe that $r(A_i \cap A_j) \geq m - 1$ since $|A_i \cap A_j| \geq m - 1$ and every set of size $m - 1$ is independent. So we have

$$r(A_i \cup A_j) + r(A_i \cap A_j) \geq m + (m - 1) > (m - 1) + (m - 1) \geq r(A_i) + r(A_j).$$

This contradicts submodularity of the rank function. ■

For any paving matroid, Lemma 29 implies that its independent sets can be written in the form

$$\mathcal{I} = \{ I : |I| \leq m \wedge |I \cap A_i| \leq m - 1 \ \forall i \},$$

where $|A_i \cap A_j| \leq m - 2$ for each $i \neq j$. This is a special case of Theorem 10 since we may apply Lemma 28 with each $b_i = m - 1$ and $\mu = m$, since

$$\min_{i, j \in [k]} (b_i + b_j - |A_i \cap A_j|) \geq 2(m - 1) - (m - 2) = m.$$

G Additional Proofs for the $O(\sqrt{n})$ -approximation algorithm

In this section we present our upper bounds for efficiently PMAC-learning two very broad families of functions. We give a PMAC-learning algorithm with approximation factor $O(n)$ for learning the family of non-negative, monotone, subadditive functions. We also give a PMAC-learning algorithm with approximation factor $O(\sqrt{n})$ for learning the class of non-negative, monotone, submodular functions.

We start with two lemmas concerning these classes of functions.

Lemma 30. *Let $f : 2^{[n]} \rightarrow \mathbb{R}_+$ be a non-negative, monotone, subadditive function. Then there exists a linear function \hat{f} such that $\hat{f}(S) \leq f(S) \leq n\hat{f}(S)$ for all $S \subseteq [n]$.*

Proof. Let $\hat{f}(S) = \frac{\sum_{i \in S} f(\{i\})}{n}$; clearly this is linear. By subadditivity, we have $\hat{f}(S) \geq \frac{f(S)}{n}$, so $f(S) \leq n\hat{f}(S)$. By monotonicity we have $f(S) \geq \max_i f(\{i\})$, so $n\hat{f}(S) \geq \sum_i f(\{i\})$. This implies $\hat{f}(S) \leq f(S)$, as desired. ■

A stronger result for the class of submodular functions was proven by Goemans et al. [21], using properties of submodular polyhedra and John's theorem on approximating centrally-symmetric convex bodies by ellipsoids [34].

Lemma 31 (Goemans et al. [21]). *Let $f : 2^{[n]} \rightarrow \mathbb{R}_+$ be a non-negative, monotone, submodular function with $f(\emptyset) = 0$. Then there exists a function \hat{f} of the form $\hat{f}(S) = \sqrt{w^T \chi(S)}$ where $w \in \mathbb{R}_+^n$ such that $\hat{f}(S) \leq f(S) \leq \sqrt{n}\hat{f}(S)$ for all $S \subseteq [n]$.*

We now use the preceding lemmas in proving our main algorithmic results.

Theorem 15. *Let \mathcal{F} be the class of non-negative, monotone, subadditive functions over $X = 2^{[n]}$. There is an algorithm that PMAC-learns \mathcal{F} with approximation factor $n + 1$. That is, for any distribution D over X , for any ϵ, δ sufficiently small, with probability $1 - \delta$, the algorithm produces a function f that approximates f^* within a multiplicative factor of $n + 1$ on a set of measure $1 - \epsilon$ with respect to D . The algorithm uses $\ell = \frac{48n}{\epsilon} \log\left(\frac{9n}{8\epsilon}\right)$ training examples and runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$.*

Proof. For technical reasons, because of the multiplicative error allowed by the PMAC-learning model, we will analyze separately the subset of the instance space where f^* is zero and the subset of the instance space

where f^* is non-zero. For convenience, let us define:

$$\mathcal{P} = \{ S : f^*(S) \neq 0 \} \quad \text{and} \quad \mathcal{Z} = \{ S : f^*(S) = 0 \}.$$

The main idea of our algorithm is to reduce our learning problem to the standard problem of learning a binary classifier (in fact, a linear separator) from i.i.d. samples in the passive, supervised learning setting [37, 59] with a slight twist in order to handle the points in \mathcal{Z} . The problem of learning a linear separator in the passive supervised learning setting is one where the instance space is \mathbb{R}^m , the samples come from some fixed and unknown distribution D' on \mathbb{R}^m , and there is a fixed but unknown target function $c^* : \mathbb{R}^m \rightarrow \{-1, +1\}$, $c^*(x) = \text{sgn}(u^\top x)$. The examples induced by D' and c^* are called *linearly separable* since there exists a vector u such that $c^*(x) = \text{sgn}(u^\top x)$.

The linear separator learning problem we reduce to is defined as follows. The instance space is \mathbb{R}^m where $m = n + 1$ and the distribution D' is defined by the following procedure for generating a sample from it. Repeatedly draw a sample $S \subseteq [n]$ from the distribution D until $f^*(S) \neq 0$. Next, flip a fair coin. The sample from D' is

$$\begin{aligned} (\chi(S), f^*(S)) & \quad (\text{if the coin is heads}) \\ (\chi(S), (n+1) \cdot f^*(S)) & \quad (\text{if the coin is tails}). \end{aligned}$$

The function c^* defining the labels is as follows: samples for which the coin was heads are labeled $+1$, and the others are labeled -1 .

We claim that the distribution over labeled examples induced by D' and c^* is linearly separable in \mathbb{R}^{n+1} . To prove this we use Lemma 30 which says that there exists a linear function $\hat{f}(S) = w^\top \chi(S)$ such that $\hat{f}(S) \leq f^*(S) \leq n \cdot \hat{f}(S)$ for all $S \subseteq [n]$. Let $u = ((n+1/2) \cdot w, -1) \in \mathbb{R}^m$. For any point x in the support of D' we have

$$\begin{aligned} x = (\chi(S), f^*(S)) & \quad \implies \quad u^\top x = (n+1/2) \cdot \hat{f}(S) - f^*(S) > 0 \\ x = (\chi(S), (n+1) \cdot f^*(S)) & \quad \implies \quad u^\top x = (n+1/2) \cdot \hat{f}(S) - (n+1) \cdot f^*(S) < 0. \end{aligned}$$

This proves the claim. Moreover, this linear function also satisfies $\hat{f}(S) = 0$ for every $S \in \mathcal{Z}$. In particular, $\hat{f}(S) = 0$ for all $S \in \mathcal{S}_0$ and moreover,

$$\hat{f}(\{j\}) = w_j = 0 \quad \text{for every } j \in \mathcal{U}_D \quad \text{where} \quad \mathcal{U}_D = \bigcup_{S_i \in \mathcal{Z}} S_i.$$

Our algorithm is now as follows. It first partitions the training set $\mathcal{S} = \{(S_1, f^*(S_1)), \dots, (S_\ell, f^*(S_\ell))\}$ into two sets \mathcal{S}_0 and $\mathcal{S}_{\neq 0}$, where \mathcal{S}_0 is the subsequence of \mathcal{S} with $f^*(S_i) = 0$, and $\mathcal{S}_{\neq 0} = \mathcal{S} \setminus \mathcal{S}_0$. For convenience, let us denote the sequence $\mathcal{S}_{\neq 0}$ as

$$\mathcal{S}_{\neq 0} = ((A_1, f^*(A_1)), \dots, (A_a, f^*(A_a))).$$

Note that a is a random variable and we can think of the sets the A_i as drawn independently from D , conditioned on belonging to \mathcal{P} . Let

$$\mathcal{U}_0 = \bigcup_{\substack{i \leq \ell \\ f^*(S_i) = 0}} S_i \quad \text{and} \quad \mathcal{L}_0 = \{ S : S \subseteq \mathcal{U}_0 \}.$$

Using $\mathcal{S}_{\neq 0}$, the algorithm then constructs a sequence $\mathcal{S}'_{\neq 0} = ((x_1, y_1), \dots, (x_a, y_a))$ of training examples for the binary classification problem. For each $1 \leq i \leq a$, let y_i be -1 or 1 , each with probability $1/2$. If $y_i = +1$ set $x_i = (\chi(A_i), f^*(A_i))$; otherwise set $x_i = (\chi(A_i), (n+1) \cdot f^*(A_i))$. The last step of our algorithm is to solve a linear program in order to find a linear separator $u = (w, -z)$ where $w \in \mathbb{R}^n$, $z \in \mathbb{R}$ consistent with the labeled examples (x_i, y_i) , $i = 1 \leq i \leq a$, with the additional constraints that $w_j = 0$ for $j \in \mathcal{U}_0$. The output hypothesis is $f(S) = \frac{1}{(n+1)z} w^\top \chi(S)$.

To prove correctness, note first that the linear program is feasible; this follows from our earlier discussion

using the facts (1) $S'_{\neq 0}$ is a set of labeled examples drawn from D' and labeled by c^* and (2) $\mathcal{U}_0 \subseteq \mathcal{U}_D$. It remains to show that f approximates the target on most of the points. Let \mathcal{Y} denote the set of points $S \in \mathcal{P}$ such that both of the points $(\chi(S), f^*(S))$ and $(\chi(S), (n+1) \cdot f^*(S))$ are correctly labeled by $\text{sgn}(u^\top x)$, the linear separator found by our algorithm. It is easy to show that the function $f(S) = \frac{1}{(n+1)z} w^\top \chi(S)$ approximates f^* to within a factor $n+1$ on all the points in the set \mathcal{Y} . To see this notice that for any point $S \in \mathcal{Y}$, we have

$$\begin{aligned} w^\top \chi(S) - z f^*(S) &> 0 \quad \text{and} \quad w^\top \chi(S) - z(n+1) f^*(S) < 0 \\ \implies \frac{1}{(n+1)z} w^\top \chi(S) &< f^*(S) < (n+1) \frac{1}{(n+1)z} w^\top \chi(S). \end{aligned}$$

So, for any point in $S \in \mathcal{Y}$, the function $f(S) = \frac{1}{(n+1)z} w^\top \chi(S)$ approximates f^* to within a factor $n+1$.

Moreover, by design the function f correctly labels as 0 all the examples in \mathcal{L}_0 . To finish the proof, we now note two important facts: for our choice of $\ell = \frac{16n}{\epsilon} \log\left(\frac{n}{\delta\epsilon}\right)$, with high probability both $\mathcal{P} \setminus \mathcal{Y}$ and $\mathcal{Z} \setminus \mathcal{L}_0$ have small measure. The fact that $\mathcal{Z} \setminus \mathcal{L}_0$ has small measure follows from an argument similar to the one in Claim 21. We now prove:

Claim 32. *If $\ell = \frac{16n}{\epsilon} \log\left(\frac{n}{\delta\epsilon}\right)$, then with probability at least $1 - 2\delta$, the set $\mathcal{P} \setminus \mathcal{Y}$ has measure at most 2ϵ under D .*

Proof. Let $q = 1 - p = \Pr_{S \sim D}[S \in \mathcal{P}]$. If $q < \epsilon$ then the claim is immediate, since \mathcal{P} has measure at most ϵ . So assume that $q \geq \epsilon$. Let $\mu = \mathbf{E}[a] = q\ell$. By assumption $\mu > 16n \log(n/\delta\epsilon) \frac{q}{\epsilon}$. Then Chernoff bounds give that

$$\Pr\left[a < 8n \log(n/\delta\epsilon) \frac{q}{\epsilon}\right] < \exp(-n \log(n/\delta) q/\epsilon) < \delta.$$

So with probability at least $1 - \delta$, we have $a \geq 8n \log(qn/\delta\epsilon) \frac{q}{\epsilon}$. By a standard sample complexity argument [59] (which we reproduce in Theorem 37 in Appendix H), with probability at least $1 - \delta$, any linear separator consistent with S' will be inconsistent with the labels on a set of measure at most ϵ/q under D' . In particular, this property holds for the linear separator c computed by the linear program. So for any set S , the conditional probability that either $(\chi(S), f^*(S))$ or $(\chi(S), (n+1) \cdot f^*(S))$ is incorrectly labeled, given that $S \in \mathcal{P}$, is at most $2\epsilon/q$. Thus

$$\Pr[S \in \mathcal{P} \wedge S \notin \mathcal{Y}] = \Pr[S \in \mathcal{P}] \cdot \Pr[S \notin \mathcal{Y} \mid S \in \mathcal{P}] \leq q \cdot (2\epsilon/q),$$

as required. \square

In summary, our algorithm produces a hypothesis f that approximates f^* to within a factor $n+1$ on the set $\mathcal{Y} \cup \mathcal{L}_\ell$. The complement of this set is $(\mathcal{Z} \setminus \mathcal{L}_\ell) \cup (\mathcal{P} \setminus \mathcal{Y})$, which has measure at most 3ϵ , with probability at least $1 - 3\delta$. \blacksquare

The preceding proof was for the class of subadditive functions. The proof for submodular functions is identical, replacing Lemma 30 with Lemma 31.

Theorem 16. *Let \mathcal{F} be the class of non-negative, monotone, submodular functions over $X = 2^{[n]}$. There is an algorithm that PMAC-learns \mathcal{F} with approximation factor $\sqrt{n+1}$. That is, for any distribution D over X , for any ϵ, δ sufficiently small, with probability $1 - \delta$, the algorithm produces a function f that approximates f^* within a multiplicative factor of $\sqrt{n+1}$ on a set of measure $1 - \epsilon$ with respect to D . The algorithm uses $\ell = \frac{48n}{\epsilon} \log\left(\frac{9n}{\delta\epsilon}\right)$ training examples and runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$.*

Proof. To learn the class of non-negative, monotone, submodular functions we apply the algorithm described in Theorem 15 with the following changes: (i) in the second step if $y_i = +1$ we set $x_i = (\chi(A_i), f^*(A_i)^2)$ and if $y_i = -1$ we set $x_i = (\chi(A_i), (n+1) \cdot f^*(A_i)^2)$; (ii) we output the function $f(S) = \sqrt{\frac{1}{(n+1)z}} w^\top \chi(S)$. See Algorithm 2. To argue correctness we use Lemma 31, which shows that, for

Algorithm 2 Algorithm for PMAC-learning the class of non-negative monotone submodular functions.

Input: A sequence of labeled training examples $\mathcal{S} = \{(S_1, f^*(S_1)), (S_2, f^*(S_2)), \dots, (S_\ell, f^*(S_\ell))\}$, where f^* is a submodular function.

- Let $\mathcal{S}_{\neq 0} = \{(A_1, f^*(A_1)), \dots, (A_a, f^*(A_a))\}$ be the subsequence of \mathcal{S} with $f^*(A_i) \neq 0 \forall i$. Let $\mathcal{S}_0 = \mathcal{S} \setminus \mathcal{S}_{\neq 0}$. Let \mathcal{U}_0 be the set of indices defined as $\mathcal{U}_0 = \bigcup_{\substack{i \leq \ell \\ f^*(S_i) = 0}} S_i$.
- For each $1 \leq i \leq a$, let y_i be the outcome of flipping a fair $\{+1, -1\}$ -valued coin, each coin flip independent of the others. Let $x_i \in \mathbb{R}^{n+1}$ be the point defined by

$$x_i = \begin{cases} (\chi(A_i), f^{*2}(A_i)) & \text{(if } y_i = +1\text{)} \\ (\chi(A_i), (n+1) \cdot f^{*2}(A_i)) & \text{(if } y_i = -1\text{)}. \end{cases}$$

- Find a linear separator $u = (w, -z) \in \mathbb{R}^{n+1}$, where $w \in \mathbb{R}^n$ and $z \in \mathbb{R}$, such that u is consistent with the labeled examples $(x_i, y_i) \forall i \in [a]$, and with the additional constraint that $w_j = 0 \forall j \in \mathcal{U}_0$.

Output: The function f defined as $f(S) = \sqrt{\frac{1}{(n+1)z}} w^\top \chi(S)$.

any $f \in \mathcal{F}$, the function f^2 can be approximated to within a factor of n by a linear function. The proof of Theorem 15 can then be applied to the family $\{f^2 : f \in \mathcal{F}\}$. \blacksquare

G.1 Extensions

The algorithm described for learning subadditive and submodular functions in the PMAC model is quite robust and it can be extended to handle more general cases as well as various forms of noise.

First, we can extend the results in Theorem 15 and Theorem 16 to the more general case where do not even assume that the target function is subadditive (or submodular), but that it is within a factor α of a subadditive (or submodular) function on every point in the instance space. Under this relaxed assumption we are able to achieve the approximation factor $\alpha \cdot (n+1)$ (or $\sqrt{\alpha \cdot (n+1)}$). Specifically:

Theorem 33. *Let \mathcal{F} be the class of non-negative, monotone, subadditive functions over $X = 2^{[n]}$ and let*

$$\mathcal{F}' = \{f : \exists g \in \mathcal{F}, g(S) \leq f(S) \leq \alpha \cdot g(S) \text{ for all } S \subseteq [n]\},$$

for some known $\alpha > 1$. There is an algorithm that PMAC-learns \mathcal{F}' with approximation factor $\alpha(n+1)$. The algorithm uses $\ell = \frac{48n}{\epsilon} \log\left(\frac{9n}{\delta\epsilon}\right)$ training examples and runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$.

Proof. By assumption, there exists $g \in \mathcal{F}$ such that $g(S) \leq f^*(S) \leq \alpha \cdot g(S)$. Combining this with Lemma 30, we get that there exists $\hat{f}(S) = w^\top \chi(S)$ such that

$$w^\top \chi(S) \leq f^*(S) \leq n \cdot \alpha \cdot w^\top \chi(S) \quad \text{for all } S \subseteq [n].$$

In order to learn the class of non-negative monotone submodular functions we apply the algorithm described in Theorem 15 with the following modifications: (1) in the second step if $y_i = +1$ we set $x_i = (\chi(S), f^*(S))$ and if $y_i = -1$ we set $x_i = (\chi(S), \alpha(n+1) \cdot f^*(S))$; (2) we output the function $f(S) = \frac{1}{\alpha(n+1)z} w^\top \chi(S)$. It is then easy to show that the distribution over labeled examples induced by D' and c^* is linearly separable in \mathbb{R}^{n+1} ; in particular, $u = (\alpha(n+1/2) \cdot w, -1) \in \mathbb{R}^{n+1}$ defines a good linear separator. The proof then proceeds as in Theorem 15. \blacksquare

Theorem 34. *Let \mathcal{F} be the class of non-negative, monotone, submodular functions over $X = 2^{[n]}$ and let*

$$\mathcal{F}' = \{f : \exists g \in \mathcal{F}, g(S) \leq f(S) \leq \alpha \cdot g(S) \text{ for all } S \subseteq [n]\},$$

for some known $\alpha > 1$. There is an algorithm that PMAC-learns \mathcal{F}' with approximation factor $\sqrt{\alpha \cdot (n+1)}$. The algorithm uses $\ell = \frac{48n}{\epsilon} \log\left(\frac{9n}{\delta\epsilon}\right)$ training examples and runs in time $\text{poly}(n, 1/\epsilon, 1/\delta)$.

We can also extend the results in Theorem 15 and Theorem 16 to the *agnostic case* where we assume that there exists a subadditive (or a submodular) function that agrees with the target on all but an η fraction of the points; note that on the η fraction of the points the target can be arbitrarily far from a subadditive (or a submodular) function. In this case we can still PMAC-learn with a polynomial number of samples $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta\epsilon}))$, but using a potentially computationally inefficient procedure.

Theorem 35. *Let \mathcal{F} be the class of non-negative, monotone, subadditive functions over $X = 2^{[n]}$. Let*

$$\mathcal{F}' = \{ f : \exists g \in \mathcal{F} \text{ s.t. } f(S) = g(S) \text{ on more than } 1 - \eta \text{ fraction of the points} \}.$$

There is an algorithm that PMAC-learns \mathcal{F}' with approximation factor $(n + 1)$. That is, for any distribution D over X , for any ϵ, δ sufficiently small, with probability $1 - \delta$, the algorithm produces a function f that approximates f^ within a multiplicative factor of $n + 1$ on a set of measure $1 - \epsilon - \eta$ with respect to D . The algorithm uses $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta\epsilon}))$ training examples.*

Proof Sketch. The proof proceeds as in Theorem 15. The main difference is that in the new feature space the best linear separator has error (fraction of mistakes) η . It is well known that even in the agnostic case the number of samples needed to learn a separator of error at most $\eta + \epsilon$ is $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta\epsilon}))$ (see Theorem 38 in Appendix H). However, it is NP-hard to minimize the number of mistakes, even approximately [23], so the resulting procedure uses a polynomial number of samples, but it is computationally inefficient. ■

Theorem 36. *Let \mathcal{F} be the class of non-negative, monotone, submodular functions over $X = 2^{[n]}$. Let*

$$\mathcal{F}' = \{ f : \exists g \in \mathcal{F} \text{ s.t. } f(S) = g(S) \text{ on more than } 1 - \eta \text{ fraction of the points} \}.$$

There is an algorithm that PMAC-learns \mathcal{F}' with approximation factor $\sqrt{n + 1}$. That is, for any distribution D over X , for any ϵ, δ sufficiently small, with probability $1 - \delta$, the algorithm produces a function f that approximates f^ within a multiplicative factor of $\sqrt{n + 1}$ on a set of measure $1 - \epsilon - \eta$ with respect to D . The algorithm uses $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta\epsilon}))$ training examples.*

H Standard Sample Complexity Results

We state here several known sample complexity bounds that were used for proving the results in Section 5 and Appendix G. See, e.g., [13, 3].

Theorem 37. *Let C be a set of functions from X to $\{-1, 1\}$ with finite VC-dimension $D \geq 1$. Let D be an arbitrary, but fixed probability distribution over X and let c^* be an arbitrary target function. For any $\epsilon, \delta > 0$, if we draw a sample from D of size $N(\epsilon, \delta) = \frac{1}{\epsilon} (4D \log(\frac{1}{\epsilon}) + 2 \log(\frac{2}{\delta}))$, then with probability $1 - \delta$, all hypotheses with error $\geq \epsilon$ are inconsistent with the data.*

Theorem 38. *Suppose that C is a set of functions from X to $\{-1, 1\}$ with finite VC-dimension $D \geq 1$. For any distribution D over X , any target function (not necessarily in C), and any $\epsilon, \delta > 0$, if we draw a sample from D of size*

$$m(\epsilon, \delta, D) = \frac{64}{\epsilon^2} \left(2D \ln \left(\frac{12}{\epsilon} \right) + \ln \left(\frac{4}{\delta} \right) \right),$$

then with probability at least $1 - \delta$, we have $|\text{err}(h) - \widehat{\text{err}}(h)| \leq \epsilon$ for all $h \in C$.