Learning Submodular Functions

Maria-Florina Balcan
Georgia Institute of Technology
School of Computer Science
ninamf@cc.gatech.edu

Nicholas J. A. Harvey
University of Waterloo
Dept. of Combinatorics and Optimization
harvey@math.uwaterloo.ca

ABSTRACT

There has been much interest in the machine learning and algorithmic game theory communities on understanding and using submodular functions. Despite this substantial interest, little is known about their learnability from data. 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 $\mathrm{poly}(n)$ 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 develop the first theoretical analysis of this problem, proving a number of important and nearly tight results. For instance, if the underlying distribution is a product distribution then we give a learning algorithm that achieves a constant-factor approximation (under some assumptions). However, for general distributions we provide a surprising $\tilde{O}(n^{1/3})$ lower bound based on a new interesting class of matroids and we also show a $O(n^{1/2})$ upper bound.

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. This construction shows unexpected extremal properties of submodular functions.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms, Theory

STOC'11, June 6–8, 2011, San Jose, California, USA.

1. INTRODUCTION

What does it mean to "learn a submodular function", and why would one be interested in doing that? To begin, we start by defining what submodular functions are. Let $[n] = \{1, \ldots, n\}$ be a ground set and let $f : 2^{[n]} \to \mathbb{R}$ 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:

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B) \quad (1.2)$$

for all $A \subseteq B \subseteq [n]$ and $i \notin B$. This inequality reflects a natural notion of "diminishing returns", which explains why submodularity has long been a topic of interest in economics [35]. Submodular functions have also been studied for decades in operations research and combinatorial optimization [13], 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 [18, 26, 6], for solving feature selection problems in graphical models [24], and for solving various clustering problems [29]. In fact, submodularity has been the topic of several tutorials and workshops at recent major conferences in machine learning [1, 25].

The Model. So what does it mean to "learn a submodular function"? Our definition comes from learning theory, where the goal of learning is to predict the future based on past observations. One successful approach to formalizing this goal is Valiant's PAC model [37]. However, the PAC model is primarily for learning *Boolean-valued functions*, such as threshold functions and low-depth circuits [37, 23]. 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 $f^* : 2^{[n]} \to \mathbb{R}_+$. 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,

$$\Pr_{x_1,x_2,\ldots\sim D}[\ \Pr_{x\sim D}[\, f(x) \le f^*(x) \le \alpha f(x)\,]\ \ge\ 1-\epsilon\ ] \ge\ 1-\delta,$$

where f is the output of the learning algorithm when given inputs $\{\,(x_i, f^*(x_i))\,\}_{i=1,2,\ldots}$ and the approximation ratio $\alpha \ge 1$ 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 [11, 9, 3].

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 [18, 40, 5].

There has been much work on methods for designing optimally priced bundles, in various models. These methods typically make two assumptions [18]. 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 corporations typically possess large amounts of data acquired from past consumer purchases [31]. 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

We prove several algorithmic results and lower bounds in the PMAC model, as well as surprising structural results on submodular functions and matroids.

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.

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 matroids with surprising extremal properties, which we describe below.

Algorithm for general distributions. Our $\tilde{\Omega}(n^{1/3})$ inapproximability result for general distributions turns out to be nearly optimal. We give an algorithm to PMAC-learn an arbitrary non-negative, monotone, submodular function with approximation factor $O(\sqrt{n})$.

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 [15]. We 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 would yield an improved analysis of our algorithm for that subclass.

A new family of extremal matroids. Our inapproximability result is based on a new family of matroids[1] with several interesting properties. The technical question we explore is: given a set family $\mathcal{A} = \{A_1, \ldots, A_k\}$ and integers $b_1, \ldots, b_k$, when is

$$\mathcal{I} \ =\ \{\ I\ :\ |I| \le r\ \ \wedge\ \ |I \cap A_i| \le b_i\ \ \forall i = 1, \ldots, k\ \} \quad (1.3)$$

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 when the A_i's form an error-correcting code of constant weight r and minimum distance 4, and each $b_i = r - 1$; this 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 by a novel and highly technical construction which gives a substantial generalization of partition and paving matroids. 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

[1] For a brief definition of matroids, see Section 2. For further discussion, we refer the reader to standard references [30, 32].

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. We provide an interesting "approximate characterization" of matroids. It is well-known that defining the function $f : 2^{[n]} \to \mathbb{R}$ by $f(S) = h(|S|)$ gives a submodular function if $h : \mathbb{R} \to \mathbb{R}$ 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 based on our concentration inequality for Lipschitz, submodular functions under product distributions.

1.2 Related Work

Learning real-valued functions and the PMAC Model. In the machine learning literature [19, 38], learning real-valued functions (in the distributional learning setting) is often addressed by considering the squared error loss[2], i.e. $\mathbf{E}_x\left[\left(f(x) - f^*(x)\right)^2\right]$. However, this does not distinguish between the case of having low error on most of the distribution and high error on just a few points, versus moderately high error everywhere. Thus, a lower bound for the squared error loss is not so meaningful. In comparison, 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. [15], 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 PMAC model falls into the more widely studied passive supervised learning setting [4, 23, 37, 38], 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. Additionally, it requires query access to the target and so does not apply to our framework. 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.

On the other hand, our lower bound is significantly more involved than the lower bound of Goemans et al. [15]. (Their lower bound was slightly improved by Svitkina and Fleischer [34].) These previous lower bounds also use matroids of the form in Eq. (1.3), although they 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*. A similar situation occurs 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 they are trivial to PAC-learn. 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] Other loss functions are also used, such as L_1 loss, but these are not substantially different from squared error loss.

[3] Technically speaking, their model can be viewed as "exact learning with value queries", which is not very natural from a machine learning perspective. In particular, in many learning applications arbitrary membership or value queries are undesirable because natural oracles, such as hired humans, have difficulty labeling synthetic examples [8]. Also, negative results for exact learning do not necessarily imply hardness for learning in more widely used learning models. We discuss this in more detail below.

2. FORMALIZING THE MODEL

2.1 Preliminaries

Notation. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. 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 $\chi(S) \in \{0,1\}^n$, where $\chi(S)_i$ is 1 if i is in S and 0 otherwise. We frequently use this natural isomorphism between $\{0,1\}^n$ and $2^{[n]}$.

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 [14, 27, 30, 32]. We will be concerned with the following properties of set functions. We say that $f : 2^{[n]} \to \mathbb{R}$ is

- *Non-negative* if $f(S) \geq 0$ for all S.
- *Monotone* (or *non-decreasing*) if $f(S) \leq f(T)$ for all $S \subseteq T$.
- *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 satisfy $f(\emptyset) = 0$.

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*. The maximal independent sets all have the same cardinality, which is the *rank* of the matroid. The *rank function* of the matroid is the function $\mathrm{rank}_{\mathbf{M}} : 2^{[n]} \to \mathbb{N}_+$ defined by

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

It is well-known that $\mathrm{rank}_{\mathbf{M}}$ 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 $f^* : \{0,1\}^n \to \mathbb{R}_+$. 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 polynomial time computation on the labeled examples $\mathcal{S}$, then must output a hypothesis function $f : \{0,1\}^n \to \mathbb{R}_+$. The

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 $(S_1, f^*(S_1)), \ldots, (S_\ell, f^*(S_\ell))$, parameters ϵ and l.

- 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 $f(A) = 1$ otherwise.

goal is that, with high probability, f is a good approximation of the target for most points in D. Formally:

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 i.i.d. from D.*
- *The number of inputs ℓ provided to $\mathcal{A}$ and the running time of $\mathcal{A}$ are both at most $\mathrm{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$.*
- *The output of $\mathcal{A}$ is a function $f : \{0,1\}^n \to \mathbb{R}$ that satisfies*

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

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. Details are given in the full version of the paper [7]. 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.

In Section 3 we also study PMAC-learnability under a *product distribution* D on $\{0,1\}^n$, meaning that if x is a sample from D, then the events $x_i = 1$ and $x_j = 1$ are independent for every $i \neq j$.

3. PRODUCT DISTRIBUTIONS

We consider learnability of submodular functions when the underlying distribution is a product distribution. All proofs are deferred to the full version of the paper.

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 1. Let D be a product distribution on $\{0,1\}^n$. For any sufficiently small $\epsilon > 0$ and $\delta > 0$, Algorithm 1 PMAC-learns $\mathcal{F}$ with approximation factor $\alpha = O(\log(1/\epsilon))$. The number of training examples used is $\ell = 10n^2\log(1/\delta) + n\log(n/\delta)/\epsilon$. If $\mathbf{E}[\, f^*(X)\,] \geq c\log(1/\epsilon)$, for sufficiently large c, then the approximation factor improves to 8.*

An important class to which Theorem 1 applies is that of matroid rank functions. Note that if the minimum non-zero value for functions in $\mathcal{F}$ is $\eta < 1$, then a simple modification yields an approximation factor of $O(\log(1/\epsilon)/\eta)$. 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. So f^* is well-approximated by the constant function that equals the empirical average. This idea is used 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 proving Theorem 1 is the strong concentration bound:

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

$$\Pr\Big[\, f(X) \leq b - t\sqrt{b}\,\Big]\cdot\Pr[\, f(X)\geq b\,] \;\leq\; \exp(-t^2/4).$$

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.[4]

Theorem 2 most naturally implies concentration around a median of $f(X)$. By standard manipulations, e.g., [21, §2.5] or [28, §20.2], this also implies concentration around the expected value. We obtain:

COROLLARY 3. *Let $f : 2^{[n]} \to \mathbb{R}_+$ 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*

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

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 quite 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 [27, pp. 251]

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

Surprisingly, we now show that a partial converse is true.

THEOREM 5. *Let $f : 2^{[n]} \to \mathbb{Z}_+$ be the rank function of a matroid with no loops, i.e., $f(S) \geq 1$ whenever $S \neq \emptyset$. Fix $\epsilon > 0$,*

[4] Our proof of Theorem 2 is based on the Talagrand inequality [2, 28]. Independently, Chekuri and Vondrák [12] proved a similar result using the FKG inequality. Concentration results of this flavor can also be proven using the framework of self-bounding functions [10], as observed in an earlier paper by Hajiaghayi et al. [17]; see also the survey by Vondrák [39].

*sufficiently small. There exists a concave function $h : [0, n] \to \mathbb{R}$ such that, for **every** $k \in [n]$, and for a $1 - \epsilon$ fraction of the sets $S \in \binom{[n]}{k}$, for some absolute constant c we have*

$$h(k)/(c\log(1/\epsilon)) \;\le\; f(S) \;\le c\log(1/\epsilon)h(k).$$

The idea behind this theorem is as follows. For $x \in [0, n]$, we define $h(x)$ to be the expected value of f under the product distribution which samples elements with probability x/n. The value of f under this distribution is tightly concentrated around $h(x)$, by the results of the previous section. For $k \in [n]$, the distribution defining $h(k)$ 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 $h(k)$. The concavity of this function h is a consequence of submodularity of f.

4. INAPPROXIMABILITY UNDER ARBITRARY DISTRIBUTIONS

The simplicity of Algorithm 1 might make one hope that a constant-factor approximation is possible under arbitrary distributions. However, the following theorem, which provides new insight into the inherent complexity of submodular functions, shows that this is not the case. A proof is in the appendix.

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 holds even for the subclass of matroid rank functions.*

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 gives a complexity-theoretic hardness result; see the full version of the paper [7].

As we described in Section 1.1, the proof of Theorem 6 proceeds by constructing a family of matroids whose rank functions take wildly varying values on a certain set of points. The high level idea is to show that for a super-polynomial sized set of k points in $\{0, 1\}^n$, for *any* partition of those points into HIGH and LOW, we can construct a matroid where the points in HIGH have rank r_{high} and the points in LOW have rank r_{low}, and the ratio $r_{\text{high}}/r_{\text{low}} = \tilde{\Omega}(n^{1/3})$. This will immediately imply hardness for learning over the uniform distribution on these k points from any polynomial-sized sample, even with value queries. Specifically, the following theorem (proven in Section 4.3) gives this construction:

THEOREM 7. *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*

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

For example, by picking $k = n^{\log n}$, 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, as promised.

4.1 Discussion of Theorem 7

To understand Theorem 7, consider the set family defined in Eq. (1.3), namely

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

If $\mathcal{I}$ is a matroid, and if $\text{rank}(A_i) = b_i$, then perhaps such a construction can be used to prove Theorem 7.

Even in the case $k = 2$, understanding $\mathcal{I}$ is quite interesting. First of all, $\mathcal{I}$ typically is not 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 are not equicardinal, which violates a basic matroid property. However, one can verify that $\mathcal{I}$ is a matroid if we require that $r \le b_1 + b_2 - |A_1 \cap A_2|$. We can even relax the constraint $|I| \le r$ to obtain

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

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

This discussion of the case $k = 2$ is quite simple. The main contribution of Theorem 7 is to generalize this discussion to the case $k > 2$. The generalization is highly technical, as Theorem 7 imposes numerous 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*. There are numerous challenges in proving the desired result. Indeed, our first construction, Theorem 8, falls short of the mark as it cannot handle the case $k > n$, whereas proving Theorem 6 requires $k = n^{\omega(1)}$. Theorem 9 improves the first construction with several ideas, and it provides the basis for proving Theorem 7 and Theorem 6.

We can also interpret Theorem 7 through the lens of the *submodular completion problem*. Suppose we have partially defined a set function $f : 2^{[n]} \to \mathbb{R}$, 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 [33]. Theorem 7 sheds some light on the submodular completion problem, for the special case when each A_i belongs to our particular family $\mathcal{A}$. Its proof shows that for *any* b_i's satisfying $8\log k \le b_i \le |A_i|$, the remaining values can be chosen such that f is submodular.

4.2 Our New Matroid Constructions

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

$$\{I \,:\, |I \cap A_1| \le b_1 \;\wedge\; |I \cap A_2| \le b_2 \;\wedge\; |I \cap (A_1 \cup A_2)| \le b_1 + b_2 - |A_1 \cap A_2|\}$$

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

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

$$\mathcal{I} \;=\; \{\, I \,:\, |I \cap A(J)| \;\le\; f(J) \;\; \forall J \subseteq [k] \,\}. \tag{4.1}$$

where the function $f : 2^{[k]} \to \mathbb{Z}$ is defined by

$$f(J) \;:=\; \sum_{j \in J} b_j \;-\; \Big(\sum_{j \in J} |A_j| - |A(J)|\Big), \tag{4.2}$$

$$\text{where } A(J) \;:=\; \bigcup_{j \in J} A_j.$$

In the definition of $f(J)$, we should think of $-\big(\sum_{j\in J}|A_j| - |A(J)|\big)$ 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 $-|A_1 \cap A_2|$, as it was in the case $k = 2$. Furthermore, if the A_j's are all disjoint then the error terms are all 0, so the family $\mathcal{I}$ reduces to $\{ I : |I \cap A_j| \le b_j \ \forall j \in [k] \}$, which is a partition matroid.

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

THEOREM 8. *The family $\mathcal{I}$ given in Eq.* (4.1) *is the family of independent sets of a matroid, if it is non-empty.*

As mentioned above, Theorem 8 does not suffice to prove Theorem 6. To see why, suppose that $k > n$ and that $b_i < |A_i|$ for every i. Then $f([k]) \le n - k < 0$, and therefore $\mathcal{I}$ is empty. So the construction of Theorem 8 is only applicable when $k \le n$, 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 $\mathcal{I}$. We will introduce an additional parameter τ, and only keep constraints for $|J| < \tau$. 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 $f : 2^{[k]} \to \mathbb{R}$ is called (μ,τ)-large if*

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

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

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

THEOREM 9. *Suppose that the function f defined in Eq.* (4.2) *is (μ,τ)-large. Then the family*

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

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

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

This construction yields quite a broad family of matroids. 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 [30].

4.3 Theorem 7 and Matroids from Lossless Expanders

To prove Theorem 7, 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*

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

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

$$|\Gamma(u)| = d \quad \forall u \in U$$
$$|\Gamma(J)| \ge (1-\epsilon)\cdot d \cdot |J| \quad \forall J \subseteq U,\ |J| \le L.$$

Lossless expanders are well-studied [20], and their existence is discussed below in Theorem 13. Given such a G, we will construct our set family $\mathcal{A} = \{A_1, \ldots, A_k\} \subseteq 2^{[n]}$ 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 7 must be reflected in the various parameters of G, so let us now make clear the relationships between these parameters.

$$U = [k], \quad V = [n], \quad d = \mu,$$
$$L = 2\tau - 2, \quad \epsilon = \frac{b}{4\mu}, \quad b \ge \frac{2\mu}{\tau}. \tag{4.3}$$

(The actual values are chosen below in Eq. (4.6).) Thus we have:

$$|A_i| = \mu \qquad \forall i \in U$$
$$|A(J)| \ge (1-\epsilon)\cdot\mu\cdot|J| \qquad \forall J \subseteq U,\ |J| \le 2\tau - 2. \tag{4.4}$$

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. (4.4)), 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 9. 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 $f_\mathcal{B} : 2^{U_\mathcal{B}} \to \mathbb{R}$, defined as in Eq. (4.2) by

$$f_\mathcal{B}(J) = \sum_{j\in J} b - \Big(\sum_{j\in J}|A_j| - |A(J)|\Big) = (b-\mu)|J| + |A(J)|.$$

CLAIM 10. *$f_\mathcal{B}$ is (μ,τ)-large.*

All claims in this section are proven in Appendix A. By this claim, Theorem 9 implies that

$$\mathcal{I}_\mathcal{B} = \Big\{ I : |I| \le \mu \wedge |I \cap A(J)| \le f_\mathcal{B}(J)\, \forall J \subseteq U_\mathcal{B},\ |J| < \tau \Big\} \tag{4.5}$$

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

The next step in proving Theorem 7 is to analyze $\mathrm{rank}_{\mathbf{M}_\mathcal{B}}(A_i)$ for $A_i \in \mathcal{A}$. This is accomplished by the following two claims, which follow from Eq. (4.4) without too much difficulty.

CLAIM 11. *Suppose that $b \le \mu$. Then for all $\mathcal{B} \subseteq \mathcal{A}$ and all $A_i \in \mathcal{B}$, we have $\mathrm{rank}_{\mathbf{M}_\mathcal{B}}(A_i) = b$, .*

CLAIM 12. *$\mathrm{rank}_{\mathbf{M}_\mathcal{B}}(A_i) = \mu$ for all $\mathcal{B} \subseteq \mathcal{A}$ and $A_i \in \mathcal{A} \setminus \mathcal{B}$.*

Lastly, we show the existence of an expander graph G with parameters that are sufficient to prove Theorem 7. The following probabilistic construction is folklore. We thank Atri Rudra for helpful discussions regarding this construction and for stating it in these general terms. A less general statement along the same lines can be found in Vadhan's survey [36, Theorem 4.4].

THEOREM 13. *Let $k \ge 2$, $\epsilon \ge 0$, $L \le k$, $d \ge 2\log(k)/\epsilon$ and $n \ge 6Ld/\epsilon$. Then a (d, L, ϵ)-lossless expander exists.*

To prove Theorem 7, we choose $k = 2^{o(n^{1/3})}$ and $\mu = |A_i| = n^{1/3}$ (cf. Eq. (4.4)). It is clear from Claim 11 that we must choose $b = 8 \log k$. (We require that k is at most $2^{o(n^{1/3})}$ because Claim 11 assumes $b \leq \mu$.) Following Eq. (4.3), we can therefore take

$$\begin{aligned} d &= \mu = n^{1/3}, \qquad \epsilon = \frac{b}{4\mu} = \frac{2\log k}{n^{1/3}}, \\ \tau &= \frac{2\mu}{b} = \frac{n^{1/3}}{4\log k}, \qquad L = 2\tau - 2 \leq \frac{n^{1/3}}{2\log k}. \end{aligned} \tag{4.6}$$

This satisfies the hypotheses of Theorem 13, so our desired expander exists and Theorem 7 is proven.

5. $O(\sqrt{n})$-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. We start with two lemmas concerning these classes of functions.

LEMMA 14. *Let $f : 2^{[n]} \to \mathbb{R}_+$ 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 $S \subseteq [n]$.*

A stronger result for the class of submodular functions was proven by Goemans et al. [15], using properties of submodular polyhedra and John's theorem on approximating centrally-symmetric convex bodies by ellipsoids [22].

LEMMA 15 (GOEMANS ET AL. [15]). *Let $f : 2^{[n]} \to \mathbb{R}_+$ be a non-negative, monotone, submodular function with $f(\emptyset) = 0$. Then there exists a function $\hat{f}$ of the form $\hat{f}(S) = \sqrt{w^{\mathsf{T}}\chi(S)}$ 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 prove our main algorithmic results.

THEOREM 16. *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 $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$.*

PROOF. Because of the multiplicative error allowed by the PMAC-learning model, we will separately analyze 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 [23, 38] 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 $c^* : \mathbb{R}^m \to \{-1, +1\}$, $c^*(x) = \mathrm{sgn}(u^{\mathsf{T}}x)$. The examples induced by D' and c^* are called *linearly separable* since there exists a vector u such that $c^*(x) = \mathrm{sgn}(u^{\mathsf{T}}x)$.

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

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

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 14 which says that there exists a linear function $\hat{f}(S) = w^{\mathsf{T}}\chi(S)$ 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: if $x = (\chi(S), f^*(S))$, then

$$u^{\mathsf{T}}x = (n + 1/2) \cdot \hat{f}(S) - f^*(S) > 0$$

and if $x = (\chi(S), (n + 1) \cdot f^*(S))$ then

$$u^{\mathsf{T}}x = (n + 1/2) \cdot \hat{f}(S) - (n + 1) \cdot f^*(S) < 0.$$

This proves the claim. The linear function $\hat{f}$ also satisfies $\hat{f}(S) = 0$ for every $S \in \mathcal{Z}$, and moreover:

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

Our algorithm is now as follows. It first partitions the training set $\mathcal{S} = \{(S_1, f^*(S_1)), \ldots, (S_\ell, f^*(S_\ell))\}$ 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

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

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

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

Using $\mathcal{S}_{\neq 0}$, we construct $\mathcal{S}'_{\neq 0} = ((x_1, y_1), \ldots, (x_a, y_a))$, a sequence 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 $f(S) = \frac{1}{(n+1)z} w^{\mathsf{T}}\chi(S)$.

To prove correctness, note first that the linear program is feasible; this follows from our earlier discussion using the facts (1) $\mathcal{S}'_{\neq 0}$ 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 $\mathrm{sgn}(u^{\mathsf{T}}x)$, the linear separator found by our algorithm. It is easy to show that the function $f(S) = \frac{1}{(n+1)z} w^{\mathsf{T}}\chi(S)$ 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

$$w^{\mathsf{T}}\chi(S) - zf^*(S) > 0 \quad \text{and} \quad w^{\mathsf{T}}\chi(S) - z(n+1)f^*(S) < 0$$
$$\implies \frac{1}{(n+1)z}w^{\mathsf{T}}\chi(S) < f^*(S) < (n+1)\frac{1}{(n+1)z}w^{\mathsf{T}}\chi(S).$$

So, for any point in $S \in \mathcal{Y}$, the function $f(S) = \frac{1}{(n+1)z}w^{\mathsf{T}}\chi(S)$ 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.

CLAIM 17. *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 [38], with probability at least $1-\delta$, any linear separator consistent with $\mathcal{S}'$ 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

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

as required. □

CLAIM 18. *If $\ell = \frac{16n}{\epsilon}\log\left(\frac{n}{\delta\epsilon}\right)$, then with probability at least $1-\delta$, the set $\mathcal{Z} \setminus \mathcal{L}_0$ has measure at most ϵ.*

PROOF. For $k \leq \ell$, define

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

So $\mathcal{L}_\ell = \mathcal{L}_0$. By subadditivity, monotonicity, and non-negativity 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 $S_{k+1}, \ldots, S_{k'}$, 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. □

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 $1-3\delta$. □

The preceding proof was for the class of subadditive functions. The proof for submodular functions is identical, replacing Lemma 14 with Lemma 15.

THEOREM 19. *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 $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$.*

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

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

- Let $\mathcal{S}_{\neq 0} = \{(A_1, f^*(A_1)), \ldots, (A_a, f^*(A_a))\}$ 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
$$x_i = \begin{cases}(\chi(A_i), f^{*2}(A_i)) & (\text{if } y_i = +1)\\ (\chi(A_i), (n+1)\cdot f^{*2}(A_i)) & (\text{if } y_i = -1).\end{cases}$$
- 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 $f(S) = \sqrt{\frac{1}{(n+1)z}w^{\mathsf{T}}\chi(S)}$.

PROOF. To learn the class of non-negative, monotone, submodular functions we apply the algorithm described in Theorem 16 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 $f(S) = \sqrt{\frac{1}{(n+1)z}w^{\mathsf{T}}\chi(S)}$. See Algorithm 2. To argue correctness we use Lemma 15, 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 16 can then be applied to the family $\{\, f^2 : f \in \mathcal{F}\,\}$. □

Remark. Our algorithm proving Theorem 19 is significantly simpler than the algorithm of Goemans et al. [15] which achieves a slightly worse approximation factor in the exact learning model with value queries.

Extensions. Our algorithms for learning submodular and subadditive functions are quite robust and can be extended to handle more general scenarios, including forms of noise. Details are given in the full version of the paper [7]

6. CONCLUSIONS AND OPEN QUESTIONS

In this work we develop the first theoretical analysis for learning submodular functions in a 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 guarantees, achieving constant-factor approximations, under natural distributional assumptions. These results provide new insights on the inherent complexity of submodular functions.

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 matroid and submodular functions, which are likely to be useful in other contexts as well.

Our work opens up a number of interesting research directions. A concrete technical question is to close the gap between the $O(n^{1/2})$ upper bound in Theorem 19 and the $\tilde{\Omega}(n^{1/3})$ lower bound in Theorem 6. We suspect that if the lower bound can be improved, then the matroids or submodular functions used in its proof are likely to be very interesting. It would also be interesting to identify subclasses of submodular functions which are PMAC-learnable with approximation ratio better than $O(\sqrt{n})$.

More generally, our PMAC model provides a new approach for analyzing the learnability of real valued functions, and it would be particularly interesting to understand the PMAC-learnability of other natural classes of real-valued functions.

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.

7. REFERENCES

[1] NIPS workshop on discrete optimization in machine learning: Submodularity, sparsity & polyhedra (DISCML), 2009. http://www.discml.cc/.
[2] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 2000.
[3] K. Amano and A. Maruoka. On learning monotone boolean functions under the uniform distribution. *TCS*, 350(5):3–12, 2006.
[4] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.
[5] Y. Bakos and E. Brynjolfsson. Bundling information goods: Pricing, profits, and efficiency. *Management Science*, 45(12), 1999.
[6] M. F. Balcan, A. Blum, and Y. Mansour. Item pricing for revenue maxmimization. In *ACM EC*, 2009.
[7] M.-F. Balcan and Nicholas J. A. Harvey. Learning submodular functions, August 2010. arXiv:1008.2159.
[8] E. Baum and K. Lang. Query learning can work poorly when a human oracle is used. In *IJCNN*, 1993.
[9] A. Blum, C. Burch, and J. Langford. On learning monotone boolean functions. In *FOCS*, 1998.
[10] S. Boucheron, G. Lugosi, and P. Massart. On concentration of self-bounding functions. *Electronic Journal of Probability*, 14:1884–1899, 2009.
[11] N. H. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747 – 770, 1996.
[12] C. Chekuri and J. Vondrák. Randomized pipage rounding for matroid polytopes and applications, September 2009. arXiv:0909.4348v1.
[13] 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.
[14] S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2005.
[15] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *SODA*, 2009.
[16] V. Guruswami, C. Umans, and S. P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *Journal of the ACM*, 56(4), 2009.
[17] M. T. Hajiaghayi, J. H. Kim, T. Leighton, and H. Räcke. Oblivious routing in directed graphs with random demands. In *STOC*, 2005.
[18] W. Hanson and R. K. Martin. Optimal bundle pricing. *Management Science*, 36(2), 1990.
[19] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009.
[20] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the AMS*, 43:439–561, 2006.
[21] S. Janson, T. Łuczak, and A. Rucíński. *Random Graphs*. Wiley-Interscience, 2000.
[22] 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.
[23] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
[24] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.
[25] A. Krause and C. Guestrin. Intelligent information gathering and submodular function optimization, 2009. http://submodularity.org/ijcai09/index.html.
[26] B. Lehmann, D. J. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55:270–296, 2006.
[27] L. Lovász. Submodular functions and convexity. *Mathematical Programmming: The State of the Art*, 1983.
[28] M. Molloy and B. Reed. *Graph Colouring and the Probabilistic Method*. Springer, 2001.
[29] M. Narasimhan and J. Bilmes. Local search for balanced submodular clusterings. In *IJCAI*, 2007.
[30] J. G. Oxley. *Matroid Theory*. Oxford University Press, 1992.
[31] P. Rusmevichientong, B. Van Roy, and P. W. Glynn. A nonparametric approach to multiproduct pricing. *Operations Research*, 54(1), 2006.
[32] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2004.
[33] C. Seshadhri and J. Vondrák. Is submodularity testable? In *ICS*, 2011.
[34] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, 2008.
[35] D. M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.
[36] S. P. Vadhan. Pseudorandomness I. *Foundations and Trends in Theoretical Computer Science*. To Appear.
[37] L.G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.
[38] V. N. Vapnik. *Statistical Learning Theory*. Wiley and Sons, 1998.
[39] J. Vondrák. A note on concentration of submodular functions, May 2010. arXiv:1005.2791.
[40] R. B. Wilson. *Nonlinear Pricing*. Oxford University Press, 1997.

APPENDIX

A. ADDITIONAL PROOFS

In this section, we will prove Theorem 6, Theorem 8 and Theorem 9. The latter two 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 $f : \mathcal{C} \to \mathbb{Z}$ be a function. Consider the family

$$\mathcal{I} = \{ I : |I \cap C| \leq f(C) \ \forall C \in \mathcal{C} \}. \tag{A.1}$$

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:

$$\begin{aligned} \forall I \in \mathcal{I}, \quad & C_1, C_2 \in T(I) \\ & \implies (C_1 \cup C_2 \in T(I)) \ \vee \ (C_1 \cap C_2 = \emptyset). \end{aligned} \tag{A.2}$$

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 20. *$\mathcal{I}$ 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 $C_1, \ldots, C_m$ 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. (A.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 $x \in I' \setminus \big(C^* \cup I\big)$. Then $I + x \in \mathcal{I}$ because for every $C \ni x$ we have $|I \cap C| \leq f(C) - 1$. □

We now use Lemma 20 to prove Theorem 8.

Theorem 8. *The family $\mathcal{I}$ defined in Eq.* (4.1)*, namely*

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

where

$$f(J) \;:=\; \sum_{j \in J} b_j - \Big(\sum_{j \in J} |A_j| - |A(J)| \Big) \quad \textit{and} \quad A(J) \;:=\; \bigcup_{j \in J} A_j,$$

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 20. 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 *super*modular. The following proof is a simplification of our original proof, due to Jan Vondrák.

PROOF. We apply Lemma 20 to $\mathcal{C} = \{\, A(J) \,:\, J \subseteq [k] \,\}$ and the function $f' : \mathcal{C} \to \mathbb{Z}$ 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 $h_I : 2^{[k]} \to \mathbb{Z}$ 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 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. 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. (A.2) holds, so the theorem follows from Lemma 20. □

A similar approach is used for our second construction.

Theorem 9. *Suppose that the function f defined in Eq.* (4.2) *is (μ, τ)-large. Then the family*

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

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 20.

Case 1: $\max\{|J_1|, |J_2|\} \geq \tau$. Assume $|J_1| \geq |J_2|$. 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 8, 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. □

Claim 10. *$f_{\mathcal{B}}$ is (μ, τ)-large.*

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

$$\begin{aligned} f_{\mathcal{B}}(J) \;&=\; (b - \mu)|J| + |A(J)| \\ &\geq\; b|J| - \epsilon\mu|J| \qquad \text{(by Eq. (4.4) and } |J| \leq 2\tau - 2) \\ &=\; \frac{3b}{4}|J| \qquad \text{(since } \epsilon = b/4\mu) \end{aligned} \tag{A.3}$$

This shows $f_{\mathcal{B}}(J) \geq 0$. If additionally $|J| \geq \tau$ then $f_{\mathcal{B}}(J) \geq \mu$ since we require $b \geq 2\mu/\tau$. So, $f_{\mathcal{B}}$ is (μ, τ)-large. □

Claim 11. *Suppose that $b \leq \mu$. Then for all $\mathcal{B} \subseteq \mathcal{A}$ and all $A_i \in \mathcal{B}$, we have $\operatorname{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i) = b$, .*

PROOF. The definition of $\mathcal{I}_{\mathcal{B}}$ includes the constraint $|I \cap A_i| \leq f_{\mathcal{B}}(\{i\}) = b$. This immediately implies $\operatorname{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i) \leq b$. 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 12. *$\operatorname{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i) = \mu$ for all $\mathcal{B} \subseteq \mathcal{A}$ and $A_i \in \mathcal{A} \setminus \mathcal{B}$.*

PROOF. Since $\mu = |A_i|$, the condition $\operatorname{rank}_{\mathbf{M}_{\mathcal{B}}}(A_i) = \mu$ 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. (4.5)).

The constraint $|A_i| \leq \mu$ is trivially satisfied, by Eq. (4.4). So it remains to show that for every $J \subseteq U_{\mathcal{B}}$ with $|J| < \tau$, we have

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

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

$$\begin{aligned} |A_i \cap A(J)| \;&=\; |A_i| + |A(J)| - |A(J + i)| \\ &\leq\; \mu + \mu|J| - (1 - \epsilon)\mu|J + i| \quad \text{(by Eq. (4.4))} \\ &=\; \frac{b\,|J + i|}{4} \;\leq\; \frac{b\,|J|}{2} \;\leq\; f_{\mathcal{B}}(J) \quad \text{(by Eq. (A.3)).} \end{aligned}$$

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

Proof (of Theorem 6). We will apply Theorem 7 with $k = 2^t$ where $t = \omega(\log n)$. Let $\mathcal{A}$ and $\mathcal{M}$ be the families constructed by Theorem 7. 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 $f^* = \operatorname{rank}_{\mathbf{M}_{\mathcal{B}}}$. 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 $8t$ and $|A|$. The set of non-training examples has measure $1 - 2^{-t + O(\log n)}$. So the expected measure of the set on which the algorithm correctly determines the rank is at most $1/2 + 2^{-t + O(\log n)}$. On the set for which the algorithm did not correctly determine the rank, its approximation factor can be no better than $n^{1/3}/(8t)$. ■