Learning Cooperative Games

Maria-Florina Balcan, Ariel D. Procaccia and Yair Zick
Carnegie Mellon University
ninamf,arielpro,yairzick@cs.cmu.edu

Abstract

This paper explores a PAC (probably approximately correct) learning model in cooperative games. Specifically, we are given m random samples of coalitions and their values, taken from some unknown cooperative game; can we predict the values of unseen coalitions? We study the PAC learnability of several well-known classes of cooperative games, such as network flow games, threshold task games, and induced subgraph games. We also establish a novel connection between PAC learnability and core stability: for games that are efficiently learnable, it is possible to find payoff divisions that are likely to be stable using a polynomial number of samples.

1 Introduction

Cooperative game theory studies the following model. We are given a set of players $N = \{1, \ldots, n\}$, and $v : 2^N \to \mathbb{R}$ is a function assigning a value to every subset (also referred to as a *coalition*) $S \subseteq N$.

The game-theoretic literature generally focuses on revenue division: suppose that players have formed the coalition N, they must now divide the revenue $v(N)$ among themselves in some reasonable manner. However, all of the standard solution concepts for cooperative games require intimate knowledge of the structure of the underlying coalitional interactions. For example, suppose that a department head wishes to divide company bonuses among her employees in a canonically stable manner using the *core* — a division such each coalition is paid (in total) at least its value. In order to do so, she must know the value that would have been generated by every single subset of her staff. How would she obtain all this information?

Indeed, it is the authors' opinion that the *information* required in order to compute cooperative solution concepts (much more than computational complexity) is a major obstacle to their widespread implementation.

Let us therefore relax our requirements. Instead of querying every single coalition value, we would like to elicit the underlying structure of coalitional interactions using a sample of m evaluations of v on subsets of N. To be more specific, let us focus on the most common learning-theoretic model: the *probably approximately correct (PAC) model* [Kearns and Vazirani, 1994]. Briefly, the PAC model studies the following problem: we are given a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ and their values $y_1, \ldots, y_m$. There

is some function f that generated these values, but it is not known to us. We are interested in finding a function f^* that, given that $\mathbf{x}_1, \ldots, \mathbf{x}_m$ were independently sampled from some distribution $\mathcal{D}$, is *very likely* ("probably") to agree with f on *most* ("approximately") points sampled from the same distribution.

Procaccia and Rosenschein [2006] provide some preliminary results on PAC learning cooperative games, focusing on *simple* games (this is a technical term, not an opinion!) — where $v(S) \in \{0, 1\}$ for every $S \subseteq N$. Their results are mostly negative, showing that simple games require an exponential number of samples in order to be properly PAC learned (with the exception of the trivial class of unanimity games). However, the decade following the publication of their work has seen an explosive growth in the number of well-understood classes of cooperative games, as well as a better understanding of the computational difficulties one faces when computing cooperative solution concepts. This is where our work comes in.

1.1 Our Contribution

We revisit the connection between learning theory and cooperative games, greatly expanding on the results of Procaccia and Rosenschein [2006].

In Section 3, we introduce a novel relaxation of the core: it is likely (but, in contrast to the classic core, not certain) that a coalition cannot improve its payoff by working alone. We show that if a game is learnable, then likely core outcomes can also be learned (in that case we say the game is *PAC stabilizable*). This result justifies our focus on learning the values of coalitions, by relating this task to the our ultimate goal of finding "good" outcomes. Interestingly, we also prove that monotone simple games are efficiently PAC stabilizable even though they are not efficiently PAC learnable.

Motivated by the foregoing connection, in Section 4 we ask whether or not classes of games are *efficiently learnable*, that is, whether there is a polynomial-time algorithm that receives a polynomial number of samples, and outputs an accurate hypothesis with high confidence. Our main results are that network flow games [Maschler *et al.*, 2013, Chapter 17.9] are efficiently learnable with path queries (but not in general), and so are threshold task games [Chalkiadakis *et al.*, 2010], and induced subgraph games [Deng and Papadimitriou, 1994]. We also study k-vector weighted voting games [Elkind *et al.*, 2009], MC nets [Ieong and Shoham, 2005], and coalitional skill games [Bachrach and Rosenschein, 2008].

1.2 Related Work

Aside from the closely related work of Procaccia and Rosenschein [2006], there are several papers that study coalitional stability in uncertain environments. Chalkiadakis and Boutilier [2004] and Li and Conitzer [2015] assume that coalition values are drawn from some unknown distribution, and we observe noisy estimates of the values. However, both papers assume full access to the cooperative game, whereas we assume that m independent samples are observed. Other works study coalitional uncertainty: coalition values are known, but agent participation is uncertain due to failures [Bachrach *et al.*, 2012a,b; Bachrach and Shah, 2013].

Our work is also related to papers on eliciting and learning combinatorial valuation functions [Zinkevich *et al.*, 2003; Lahaie and Parkes, 2004; Lahaie *et al.*, 2005; Balcan and Harvey, 2011; Balcan *et al.*, 2012; Badanidiyuru *et al.*, 2012]. A player's valuation function in a combinatorial auction is similar to a cooperative game: it assign a value to every subset of items (instead of every subset of players). This connection allows us to draw on some of the insights from these papers. For example, as we explain below, learnability results for XOS valuations [Balcan *et al.*, 2012] informed our results on network flow games.

2 Preliminaries

2.1 Cooperative Games

A cooperative game is a tuple $\mathcal{G} = \langle N, v \rangle$, where $N = \{1, \ldots, n\}$ is a set of players, and $v : 2^n \to \mathbb{R}$ is called the *characteristic function* of $\mathcal{G}$. When the player set N is obvious, we will identify $\mathcal{G}$ with the characteristic function v, referring to v as the game. A game $\mathcal{G}$ is called *simple* if $v(S) \in \{0, 1\}$ for all $S \subseteq N$; $\mathcal{G}$ is called *monotone* if $v(S) \leq v(T)$ whenever $S \subseteq T$. One of the main objectives in cooperative games is finding "good" ways of dividing revenue: it is assumed that players have generated the revenue $v(N)$, and must find a way of splitting it. An *imputation* for $\mathcal{G}$ is a vector $\mathbf{x} \in \mathbb{R}^n$ that satisfies *efficiency*: $\sum_{i=1}^{n} x_i = v(N)$, and *individual rationality*: $x_i \geq v(\{i\})$ for every $i \in N$. The set of imputations, denoted $I(\mathcal{G})$, is the set of all possible "reasonable" payoff divisions among the players. Given a game $\mathcal{G}$, the *core* of $\mathcal{G}$ is given by

$$Core(\mathcal{G}) = \{\mathbf{x} \in I(\mathcal{G}) \mid \forall S \subseteq N : x(S) \geq v(S)\}.$$

The core is the set of all *stable* imputations: no subset of players S can deviate from an imputation $\mathbf{x} \in Core(\mathcal{G})$ while guaranteeing that every $i \in S$ receives at least as much as it gets under $\mathbf{x}$.

2.2 PAC Learning

We provide a brief overview of the PAC learning model; for a far more detailed exposition, see [Kearns and Vazirani, 1994; Shashua, 2009]. PAC learning pertains to the study of the following problem: we are interested in learning an unknown function $f : 2^N \to \mathbb{R}$. In order to estimate the value of f, we are given m samples $(S_1, v_1), \ldots, (S_m, v_m)$, where $v_j = f(S_j)$. Without any additional information, one could make arbitrary guesses as to the possible identity of f; for example, we could very well guess that $f^*(S_j) = v_j$ for all $j \in [m]$, and 0 everywhere else. Thus, in order to obtain meaningful results, we must make further assumptions. First, we restrict f to be a function from a certain class of functions $\mathcal{C}$: for example, we may know that f is a linear function of the form $f(S) = \sum_{i \in S} w_i$, but we do not know the values $w_1, \ldots, w_n$. Second, we assume that there is some distribution $\mathcal{D}$ over 2^N such that $S_1, \ldots, S_m$ were sampled i.i.d. from $\mathcal{D}$. Finally, we require that the estimate that we provide has low error over sets sampled from $\mathcal{D}$.

Formally, we are given a function $v : 2^N \to \mathbb{R}_+$, and two values $\varepsilon > 0$ (the accuracy parameter) and $\delta > 0$ (the confidence parameter). An algorithm $\mathcal{A}$ takes as input ε, δ and m samples, $(S_1, v(S_1)), \ldots, (S_m, v(S_m))$, taken i.i.d. from a distribution $\mathcal{D}$. We say that $\mathcal{A}$ can properly learn a function $f \in \mathcal{C}$ from a class of functions $\mathcal{C}$ ($\mathcal{C}$ is sometimes referred to as the *hypothesis class*), if by observing m samples — where m can depend only on n (the representation size), $\frac{1}{\varepsilon}$, and $\frac{1}{\delta}$ — it outputs a function $f^* \in \mathcal{C}$ such that with probability at least $1 - \delta$,

$$\Pr_{S \sim \mathcal{D}}[f(S) \neq f^*(S)] < \varepsilon.$$

The confidence parameter δ indicates that there is some chance that $\mathcal{A}$ will output a bad guess (intuitively, that the m samples given to the algorithm are not representative of the overall behavior of f over the distribution $\mathcal{D}$), but this is unlikely. The accuracy parameter ε indicates that for most sets sampled from $\mathcal{D}$, f^* will correctly guess the value of S.

Note that the algorithm $\mathcal{A}$ does not know $\mathcal{D}$; that is, the only thing required for PAC learnability to hold is that the input samples independent, and that future observations are also sampled from $\mathcal{D}$. In this paper, we only discuss *proper learning*; that is, learning a function $f \in \mathcal{C}$ using only functions from $\mathcal{C}$.

We say that a finite class of functions $\mathcal{C}$ is *efficiently PAC learnable* if the PAC learning algorithm described above runs in polynomial time, and its sample complexity m is polynomial in n, $\frac{1}{\varepsilon}$, and $\frac{1}{\delta}$.

Efficient PAC learnability can be established via the existence of *consistent algorithms*. Given a class of functions $\mathcal{C}$ from 2^N to $\mathbb{R}$, suppose that there is some efficient algorithm $\mathcal{A}$ that for any set of samples $(S_j, v_j)_{j=1}^m$ is able to output a function $f^* \in \mathcal{C}$ such that $f^*(S_j) = v_j$ for all $j \in [m]$, or determine that no such function exists. Then $\mathcal{A}$ is an algorithm that can efficiently PAC learn $\mathcal{C}$ given $m \geq \frac{1}{\varepsilon} \log \frac{|\mathcal{C}|}{\delta}$ samples. Conversely, if no efficient algorithm exists, then f cannot be efficiently PAC learned from $\mathcal{C}$.

To conclude, in order for a class $\mathcal{C}$ to be efficiently PAC learnable, we must have polynomial bounds on the *sample complexity* — i.e. the number of samples required in order to obtain a good estimate of functions in $\mathcal{C}$ — as well as a poly-time algorithm that finds a function in $\mathcal{C}$ which is a perfect match for the samples. We observe that in many of the settings described in this paper, the sample complexity is low, but finding consistent functions in $\mathcal{C}$ is computationally intractable (it would entail that $P = NP$ or that $NP = RP$). In contrast, the result of Procaccia and Rosenschein [2006] establishes lower bounds on the sample complexity for PAC learning monotone simple games, but there exists a simple algorithm that outputs a hypothesis consistent with any sample.

When the hypothesis class $\mathcal{C}$ is finite, it suffices to show that $\log |\mathcal{C}|$ is bounded by a polynomial in order to establish a polynomial sample complexity. In the case of an infinite class of hypotheses, this bound becomes meaningless, and other measures must be used. When learning a function that takes values in $\{0, 1\}$, the VC dimension [Kearns and Vazirani, 1994] captures the learnability of $\mathcal{C}$. Given a class $\mathcal{C}$, and a list $\mathcal{S}$ of m sets $S_1, \ldots, S_m$, we say that $\mathcal{C}$ *shatters* $\mathcal{S}$ if for every $\mathbf{b} \in \{0, 1\}^m$ there

exists some $v_{\mathbf{b}} \in \mathcal{C}$ such that $v(\mathbf{b})(S_j) = b_j$ for all j. We write

$$VCdim(\mathcal{C}) = \max\{m \mid \exists \mathcal{S}, |\mathcal{S}| = m, \mathcal{C} \text{ can shatter } \mathcal{S}\}.$$

When learning hypotheses that output real numbers (as opposed to functions that take on values in $\{0,1\}$), the notion of *pseudo dimension* is used in order to bound the complexity of a function class. Given a sample of m sets $\mathcal{S} = S_1, \ldots, S_m \subseteq N$, we say that a class $\mathcal{C}$ *shatters* $\mathcal{S}$ if there exist thresholds $r_1, \ldots, r_m \in \mathbb{R}$ such that for every $\mathbf{b} \in \{0,1\}^m$ there exists some $v_{\mathbf{b}} \in \mathcal{C}$ such that $v_{\mathbf{b}}(S_j) \geq r_j$ if $b_j = 1$, and $v_{\mathbf{b}}(S_j) < r_j$ if $b_j = 0$. We write

$$Pdim(\mathcal{C}) = \max\{m \mid \exists \mathcal{S} : |\mathcal{S}| = m, \mathcal{C} \text{ can shatter } \mathcal{S}\}.$$

It is known [Anthony and Bartlett, 2009] that if $Pdim(\mathcal{C})$ is polynomial, then the sample complexity of $\mathcal{C}$ is polynomial as well.

3 PAC Stability

In the context of cooperative games, one could think of PAC learning as the following process. A central authority wishes to find a stable outcome, but lacks information about agents' abilities. It solicits the independent valuations of m subsets of agents, and outputs an outcome that, with probability $1 - \delta$, is likely to be stable against any unknown valuations.

More formally, given $\varepsilon \in (0,1)$, we say that an imputation $\mathbf{x} \in I(\mathcal{G})$ is *ε-probably stable* under $\mathcal{D}$ if

$$\Pr_{S \sim \mathcal{D}}[x(S) \geq v(S)] \geq 1 - \varepsilon.$$

An algorithm $\mathcal{A}$ can *PAC stabilize* a class of functions $\mathcal{C}$ from 2^N to $\mathbb{R}$ if, given $\varepsilon, \delta \in (0,1)$, and m i.i.d. samples $(S_1, v(S_1)), \ldots, (S_m, v(S_m))$ of some $v \in \mathcal{C}$, with probability $1 - \delta$, $\mathcal{A}$ outputs an outcome $\mathbf{x}$ that is ε-probably stable under $\mathcal{D}$, or outputs that the core is empty. If m is polynomial in n, $\frac{1}{\varepsilon}$ and $\log \frac{1}{\delta}$, and $\mathcal{A}$ runs in polynomial time, we say that $\mathcal{C}$ is efficiently PAC stabilizable.

There is an immediate relation between PAC learnable and PAC stabilizable function classes:

Proposition 3.1. *Let $v, v^* : 2^N \to \mathbb{R}_+$ be two functions such that $\Pr_{S \sim \mathcal{D}}[v(S) = v^*(S)] \geq 1 - \varepsilon$; if $\mathbf{x} \in Core(v^*)$, then $\mathbf{x}$ is ε-probably stable under $\mathcal{D}$ for v.*

Proof. Suppose that v^* is a function that satisfies $\Pr_{S \sim \mathcal{D}}[v(S) = v^*(S)] \geq 1 - \varepsilon$, and suppose that $\mathbf{x} \in Core(v^*)$. Then,

$$\begin{aligned}
&\Pr_{S \sim \mathcal{D}}[x(S) \geq v(S)] \\
&\geq \Pr_{S \sim \mathcal{D}}[x(S) \geq v(S) \mid v(S) = v^*(S)] \cdot \Pr_{S \sim \mathcal{D}}[v(S) = v^*(S)] \\
&\geq 1 \cdot (1 - \varepsilon) = 1 - \varepsilon
\end{aligned}$$

□

We mention that Proposition 3.1 says nothing about the efficiency of finding probably stable outcomes. In order to find a PAC-stable outcome using a PAC learned function, it is essential that v belong to a class of functions that can be learned in polynomial time, and for which a core outcome can be found in polynomial time.

Moreover, there is an important subtlety here. Let $\mathcal{G}^* = \langle N, v^* \rangle$ be the PAC learned hypothesis of $\mathcal{G} = \langle N, v \rangle$. Proposition 3.1 states that if $\mathbf{x} \in Core(\mathcal{G}^*)$, then the probability that $\mathbf{x}$ violates a core constraint in $\mathcal{G}$ is small. However, there are two potential risks: first, it is possible that $Core(\mathcal{G}^*) = \emptyset$, but $Core(\mathcal{G}) \neq \emptyset$. This is not a concern if the learned hypothesis is guaranteed to have a non-empty core, or if $Core(\mathcal{G}^*)$ contains $Core(\mathcal{G})$.

Second, even if $Core(\mathcal{G}^*) \neq \emptyset$, we are not guaranteed that $\mathbf{x} \in Core(\mathcal{G}^*)$ is a valid payoff division for $\mathcal{G}$, if $v^*(N) \neq v(N)$. In our motivating setting, we assume that $v(N)$ is known, so the latter is not a major concern.

These issues do not arise if $v^*(S) \leq v(S)$ for all $S \subseteq N$. In that case, $Core(\mathcal{G}^*)$ contains $Core(\mathcal{G})$; thus, if the latter is non-empty, so is the former.

While it may be generally hard to find a core outcome for a cooperative game, it is easy to do so for *monotone simple games*, where the core has a very simple characterization (see e.g. [Maschler *et al.*, 2013, Chapter 17]) via *veto players*. We say that a player $i \in N$ is a veto player if it belongs to all winning coalitions; in other words, if $v(S) = 1$, then $i \in S$.

Fact 3.2. *Let $\mathcal{G} = \langle N, v \rangle$ be a monotone simple game, and let $V \subseteq N$ be the set of veto players for $\mathcal{G}$. If $V = \emptyset$ then $Core(\mathcal{G}) = \emptyset$; otherwise, $Core(\mathcal{G})$ consists of all imputations that assign a payoff of 0 to any $i \in N \setminus V$; in particular, if $\mathbf{x} \in Core(\mathcal{G})$ then $\sum_{i \in V} x_i = 1$.*

Leveraging Fact 3.2, we obtain the following result.

Theorem 3.3. *The class of monotone simple games is efficiently PAC stabilizable.*

The theorem is especially interesting because the class of monotone simple games is *not* efficiently PAC learnable [Procaccia and Rosenschein, 2006].

Proof of Theorem 3.3. First, if our samples contain two disjoint winning coalitions, then the core of $\mathcal{G}$ is surely empty, and we report that the core is empty. Thus, let us assume that all winning coalitions intersect. This implies that our input samples correspond to an input from a *unanimity game* [Maschler *et al.*, 2013].

A unanimity game $\mathcal{U}_V = \langle N, u_V \rangle$, has $u_V(S) = 1$ if and only if $V \subseteq S$. According to Fact 3.2,

$$Core(\mathcal{G}) = Core(\mathcal{U}_V) = \left\{ \mathbf{x} \in \mathbb{R}^n_+ \;\middle|\; \sum_{i \in V} x_i = 1 \right\}.$$

Thus, finding a probably stable outcome for $\mathcal{G}$ amounts to finding a probably stable outcome for $\mathcal{U}_V$. Procaccia and Rosenschein [2006] show that unanimity games can be efficiently PAC learned; thus, according to Proposition 3.1, unanimity games are PAC stabilizable. Moreover, deciding whether the core of a monotone simple game can be done in polynomial time (simply decide whether i is a veto player by checking

whether $v(N \setminus \{i\}) = 1$), so we can easily identify the set of veto players in the learned game, and find $\mathbf{x}$ such that

$$\begin{aligned} 1 - \varepsilon &\leq \Pr_{S \sim \mathcal{D}} [x(S) \geq u_V(S)] \\ &= \Pr_{S \sim \mathcal{D}} [x(S) \geq 1 \wedge V \subseteq S] + \Pr_{S \sim \mathcal{D}} [x(S) \geq 0 \wedge V \not\subseteq S] \\ &= \Pr_{S \sim \mathcal{D}} [x(S) \geq 1 \mid V \subseteq S] \cdot \Pr_{S \sim \mathcal{D}} [V \subseteq S] + \Pr_{S \sim \mathcal{D}} [V \not\subseteq S]. \end{aligned}$$

But this means that $\mathbf{x}$ is also ε-probably stable with respect to $\mathcal{D}$ and $\mathcal{G}$, because for every $S \subseteq N$, $v(S) = 1$ implies that $V \subseteq S$.

□

4 PAC Learnability of Common Classes of Cooperative Games

Theorem 3.3 shows that even when a class $\mathcal{C}$ is not PAC learnable using a polynomial number of samples, it is still possible to PAC stabilize it. In what follows, we explore both PAC learnability and PAC stability in common classes of cooperative games. We show when can one leverage efficient PAC learnability in order to obtain PAC stability, and identify cases where this is not possible. Some of our computational intractability results depend on the assumption that $NP \neq RP$, where RP is the class of all languages for which there exists a poly-time algorithm that for every instance I, outputs "no" if I is a no instance, and "yes" with probability $\geq \frac{1}{2}$ if it is a "yes" instance. It is believed that $NP \neq RP$ [Hemaspaandra and Ogihara, 2002].

4.1 Network Flow Games

A *network flow game* is given by a weighted, directed graph $\Gamma = \langle V, E \rangle$, with $w : E \to \mathbb{R}_+$ being the weight function for the edges. Here, $N = E$, and $v(S) = \mathit{flow}(\Gamma|_S, w, s, t)$, where flow denotes the maximum s-t flow through Γ, where edge weights are given by w, and $s, t \in V$.

We begin by showing that a similar class of functions is not efficiently learnable. We define the following family of functions, called *min-sum* functions which are defined as follows: there exists a list of n-dimensional, non-negative integer vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$. For every $S \subseteq N$, $f(S) = \min_{\ell \in [k]} \mathbf{w}_\ell(S)$, where $\mathbf{w}_\ell(S) = \sum_{j \in S} w_{\ell j}$. If $k = 1$, we say that the min-sum function is *trivial*. We note that Balcan *et al.* [2012] study the learnability of XOS valuations, where the $\min$ is replaced with a $\max$.

We define k-min-sum to be the class of min-sum functions defined with k vectors.

Lemma 4.1. *The class of k-min-sum functions is not efficiently PAC learnable unless $NP = RP$ whenever $k \geq 3$.*

Proof. Our proof relies on the fact that CNF formulas with more than two clauses are not efficiently learnable unless $NP = RP$ Pitt and Valiant [1988].

Given a set of variables $x_1, \ldots, x_n$, let us define a set of players

$$N = \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, y\}.$$

Given a k-clause CNF formula of the form $\phi = \bigwedge_{\ell=1}^{k} C_\ell$, where C_ℓ is a disjunctive clause containing literals from N (excluding the variable y), we define the following $k+1$-min-sum function $f_\phi : 2^N \to \{0,1\}$:

$$f_\phi(S) = \min\{(|C_j \cap S|)_{j=1}^m, |S \cap \{y\}|\}.$$

In order to have a value of 1, S must intersect with every C_j on at least one player; otherwise, $f_\phi(S) = 0$. Moreover, S must contain y. We note that it is possible that one cannot generate a truth assignment for ϕ from all sets S for which $f_\phi(S) = 1$; for example, $f_\phi(N) = 1$, but this is completely uninformative. Given a truth assignment T for ϕ, we define its set equivalent to be S_T, where S_T contains x_i if x_i is true in T, otherwise S_T contains $\bar{x}_i$. Also, S_T contains y. Thus, $f_\phi(S_T) = 1$ if and only if T satisfies ϕ.

Now, given a set of inputs from ϕ $(T_1, \phi(T_1)), \ldots, (T_m, \phi(T_m))$, we write $\mathcal{T} = \{T_1, \ldots, T_m\}$. For every $T \in \mathcal{T}$, we add to the input $(S_T, \phi(T))$, and $(S_T \setminus \{y\}, 0)$. The sampled point $(S_T \setminus \{y\}, 0)$ is added to ensure that the "importance" of y in the definition of f is noted by any algorithm that is consistent with the input. In other words, we can "pretend" that the input of truth assignments to ϕ is an input of an unknown $k+1$-min-sum function as defined above, and use it to define a CNF formula that is consistent with ϕ.

Suppose that there exists some consistent algorithm $\mathcal{A}$ for $k+1$-min-sum functions; that is, given a list of m inputs from an unknown $k+1$-min-sum function f, $\mathcal{A}$ outputs a list of non-negative vectors $\mathbf{w}_1^*, \ldots, \mathbf{w}_{k+1}^*$ that define a $k+1$-min-sum function that is consistent on the inputs. Then, if we input to this algorithm the inputs defined above, it will output f^* defined by $\mathbf{w}_1^*, \ldots, \mathbf{w}_{k+1}^*$, such that $f^*(S_{T_\ell}) = 1$ whenever T_ℓ is a truth assignment for ϕ. We need to show how to reconstruct a CNF formula with at most k clauses, such that $\phi^*(T) = \phi(T)$ for all $T \in \mathcal{T}$.

Let us define $\mathcal{T}_-$ to be the set of truth assignments in $\mathcal{T}$ that do not satisfy ϕ and $\mathcal{T}_+$ to be the set of truth assignments in $\mathcal{T}$ that do. Then, for every $T \in \mathcal{T}_-$, $f(S_T) = 0$, and in particular $f^*(S_T) = 0$ as well. Suppose that one of the vectors that define f^*, say $\mathbf{w}_\ell^*$, has a positive value assigned to y. In that case, $w_\ell^*(S_T) > 0$, and in particular there must be some other vector $\mathbf{w}_{\ell'}^*$ such that $w_{\ell'}^*(S_T) = 0$; in particular, $\mathbf{w}_{\ell'}^*$ has the weight of y set to 0. Suppose that all of the vectors $\mathbf{w}_1^*, \ldots, \mathbf{w}_{k+1}^*$ have the value of y set to 0; then for any truth assignment $T \in \mathcal{T}$ that satisfies ϕ, we have $(S_T, 1)$ and $(S_T \setminus \{y\}, 0)$; but since none of the weight vectors of f^* assign a positive weight to y, $f^*(S_T) = f^*(S_T \setminus \{y\})$, a contradiction to the consistency of f^*. We conclude that there exists at least one vector $\mathbf{w}_\ell^*$ that has a positive weight assigned to y.

Let us take the set of vectors who assign a weight of 0 to y, call that set $\mathcal{W}^*$; since there exist some vectors that assign a positive weight to y, $|\mathcal{W}^*| \leq k$. For every $\mathbf{w}_\ell^* \in \mathcal{W}^*$, we define a clause C_ℓ^* such that C_ℓ^* is a disjunction of all literals in the support of $\mathbf{w}_\ell^*$; in other words, if the weight of x_i in $\mathbf{w}_\ell^*$ is positive, then x_i is in C_ℓ^*, and if the weight of $\bar{x}_i$ is positive then $\bar{x}_i$ is in C_ℓ^*. Let us write the resulting CNF formula to be ϕ^*. First, since $|\mathcal{W}^*| \leq k$, the number of clauses in ϕ^* is at most k. Second, suppose that $\phi(T) = 1$, then $f^*(S_T) = 1$, and in particular, $w_\ell^*(S_T) > 0$ for all $\mathbf{w}_\ell^* \in \mathcal{W}^*$; thus, $\phi^*(T) = 1$ as well. Finally, if $\phi(T) = 0$, then $f(S_T) = 0$, and there is at least one weight vector in $\mathcal{W}^*$ that has $w_\ell^*(S_T) = 0$. The corresponding

clause C^*_ℓ is not satisfied by T, and in particular, $\phi^*(T) = 0$. This concludes the proof. □

Theorem 4.2. *Network flow functions are not efficiently learnable unless $NP = RP$.*

Proof. Our proof reduces the problem of learning min-sum functions to the problem of learning network flow functions. Given a min-sum target function f, defined by $\mathbf{w}_1, \ldots, \mathbf{w}_k$, and a distribution $\mathcal{D}$ over samples of N, we construct the directed graph $\Gamma = \langle V, E \rangle$ as follows.

For every weight vector $\mathbf{w}_\ell = (w_{1\ell}, \ldots, w_{n\ell})$, we define vertices $\ell, \ell + 1$, and n edges from ℓ to $\ell + 1$, where the capacity of the edge $e_{i\ell}$ is $w_{i\ell}$. Finally, we denote the vertex $k+1$ as the target t, and the vertex 1 as the source s. Given a set $S \subseteq N$, we write $E_S = \{e_{i\ell} \mid \ell \in [k], i \in S\}$. We observe that the flow from s to t in the constructed graph using only edges in E_S equals $f(S)$; in other words, $\mathit{flow}_\Gamma(E_S) = f(S)$ for all $S \subseteq N$. Now, given a probability distribution $\mathcal{D}$ over 2^N, we define a probability distribution over E as follows: $\Pr_{\mathcal{D}'}[E_S] = \Pr_{\mathcal{D}}[S]$ for all $S \subseteq N$, and is 0 for all other subsets of E.

We conclude that efficiently PAC learning flow_Γ under the distribution $\mathcal{D}'$ is equivalent to PAC learning f, which cannot be done efficiently by Lemma 4.1. □

Learning network flow games is thus generally a difficult task. In order to obtain some notions of tractability, let us study a variant of network flow games, where we limit our attention to sets that constitute paths in Γ. In other words, we limit our attention to distributions $\mathcal{D}$ such that if $\mathcal{D}$ assigns some positive probability to a set S, then S must be an s-t path in Γ. One natural example of such a distribution is the following: we make graph queries on Γ by performing a random walk on Γ until we either reach t or have traversed more than $|V|$ vertices.

Given a directed path $p = (w_1, \ldots, w_k)$, we let $w(p)$ be the flow that can pass through p; that is, $w(p) = \min_{e \in p} w_e$.

Theorem 4.3. *Network flow games are efficiently PAC learnable if we limit $\mathcal{D}$ to be a distribution over paths in Γ.*

Proof. Given an input $((p_1, v_1), \ldots, (p_m, v_m))$, we let $\bar{w}_e = \max_{j : e \in p_j} v_j$.

We observe that the weights $(\bar{w}_e)_{e \in E}$ are such that $\bar{w}(p_j) = w(p_j)$ for all $j \in [m]$. This is because for any $e \in p_j$, $\bar{w}_e \geq v_j$, so $\min_{e \in p_j} \bar{w}_e \geq v_j$. On the other hand, $\bar{w}_e \leq w_e$ for all $e \in E$, since $w_e \geq v_j$ for all v_j such that $e \in p_j$, and in particular $w_e \geq \max_{j : e \in p_j} v_j = \bar{w}_e$. Thus, $\min_{e \in p_j} \bar{w}_e \leq \min_{e \in p_j} w_e = v_j$. In other words, by simply taking edge weights to be the maximum flow that passes through them in the samples, we obtain a graph that is consistent with the sample in polynomial time.

Now, suppose that the set of weights on the edges of the graph according to the target weights w_e is given by $\{a_1, \ldots, a_k\}$, where $k \leq n$. Then there are $(k+1)^n \leq (n+1)^n$ possible ways of assigning values $(\bar{w}_e)_{e \in E}$ to the edges in E. In other words, there are at most $(n+1)^n$ possible hypotheses to test. Thus, in order to (ε, δ)-learn $(w_e)_{e \in E}$, where the hypothesis class $\mathcal{C}$ is of size $\leq (n+1)^n$, we need a number of samples polynomial in $\frac{1}{\varepsilon}, \log \frac{1}{\delta}$ and $\log |\mathcal{C}| \in \mathcal{O}(n \log n)$. □

Corollary 4.4. *Network flow games are efficiently PAC stabilizable if we limit $\mathcal{D}$ to be a distribution over paths in Γ.*

Proof. Theorem 4.3 establishes that if one is limited to path queries, network flow games are efficiently PAC learnable. In order to prove PAC stabilizability, we need to show that core outcomes can be found in polynomial time.

It is well-known that computing core outcomes in network flow games is easy: given an edge set $C \subseteq E$ that is a minimum cut in the graph, pay each $e \in C$ an amount equal to the flow that passes through it. We conclude that finding probably stable outcomes for network flow games can be done in polynomial time if we limit ourselves to path queries on Γ. □

4.2 Threshold Task Games

In *Threshold task games (TTG)* each player $i \in N$ has an integer weight w_i; there is a finite list of tasks $\mathcal{T}$, and each task $t \in \mathcal{T}$ is associated with a threshold $q(t)$ and a payoff $V(t)$. Given a coalition $S \subseteq N$, we let $\mathcal{T}|_S = \{t \in \mathcal{T} \mid q(t) \leq w(S)\}$. The value of S is given by $v(S) = \max_{t \in \mathcal{T}|_S} V(t)$. In other words, $v(S)$ is the value of the most valuable task that S can accomplish. Weighted voting games (WVGs) are the special case of TTGs with a single task, whose value is 1; that is, they describe linear classifiers.

Without loss of generality we can assume that all tasks in $\mathcal{T}$ have strictly monotone thresholds and values: if $q(t) > q(t')$ then $V(t) > V(t')$. Otherwise, we will have some redundant tasks. For ease of exposition, we assume that there is some task whose value is 0 and whose threshold is 0. Let $\mathcal{C}_{ttg}(Q)$ be the class of k-TTGs for which the set of task values $Q \subseteq \mathbb{R}$ of size k. The first step of our proof is to show that $\mathcal{C}^k_{ttg}(Q)$ is PAC learnable.

Lemma 4.5. *The class $\mathcal{C}^k_{ttg}(Q)$ is PAC learnable*

Proof. In order to show this, we first bound the sample complexity of $\mathcal{C}^k_{ttg}(Q)$. We claim that $Pdim(\mathcal{C}^k_{ttg}(Q)) < (k+1)(n+2)$. The proof relies on the fact that the VC dimension of linear functions is $n+1$.

Assume by contradiction that there exists some $\mathcal{S}$ of size L, where $L = (k+1)(n+2)$, and some values $r_1, \ldots, r_L \in \mathbb{R}_+$ such that for all $\mathbf{b} \in \{0,1\}^L$ there is some TTG $f_{\mathbf{b}} \in \mathcal{C}^k_{ttg}(Q)$ such that $f_{\mathbf{b}}(S_j) \geq r_j$ when $b_j = 1$, and $f_{\mathbf{b}}(S_j) < r_j$ when $b_j = 0$. We assume that $0 \leq r_1 \leq \cdots \leq r_L$. Let us write the task set to $\mathcal{T} = \{t_1, \ldots, t_k\}$, each with a value $V(t_\ell) \in Q$; we order our tasks by increasing value. Now, by the pigeonhole principle, there exists some task t_ℓ and some j^* such that $r_{j^*}, \ldots, r_{j^*+(n+1)} \in [V(t_\ell), V(t_{\ell+1}))$. In particular, if we write $\mathcal{S}^* = \{S_{j^*}, \ldots, S_{j^*+(n+1)}\}$, then for every $\mathbf{b} \in \{0,1\}^L$, there is some $f_{\mathbf{b}} \in \mathcal{C}^k_{ttg}(Q)$ (defined by an agent weight vector $\mathbf{w_b}$, and task thresholds $T^{\mathbf{b}}_1, \ldots, T^{\mathbf{b}}_k$), such that for all $S_j \in \mathcal{S}^*$, if $f_{\mathbf{b}}(S_j) > r_j$ it must be that $f_{\mathbf{b}}(S_j) > V_\ell$, i.e., $\mathbf{w_b}(S_j) > T^{\mathbf{b}}_\ell$. If $f_{\mathbf{b}}(S_j) \leq r_j$ then $\mathbf{w_b}(S_j) \leq T^{\mathbf{b}}_\ell$. Thus, $(\langle \mathbf{w_b}, T^{\mathbf{b}}_\ell \rangle)_{\mathbf{b}}$ is a set of n-dimensional linear classifiers that is able to shatter a set of size $n+2$, a contradiction. To conclude, $Pdim(\mathcal{C}^k_{ttg}(Q)) \leq (k+1)(n+2)$, which implies that the sample complexity for PAC learning TTGs is polynomial.

It is easy to construct an efficient algorithm that is consistent with any sample from $\mathcal{C}^k_{ttg}(Q)$ via linear programming.

Given the inputs,

$$(S_1, v_1), \ldots, (S_m, v_m),$$

let us write the distinct values $\alpha_1, \ldots, \alpha_\ell$ in $v_1, \ldots, v_m$, and create ℓ tasks with values $V(t_1) = \alpha_1, \ldots, V(t_\ell) = \alpha_\ell$. We observe that since $(S_1, v_1), \ldots, (S_m, v_m)$ represent outputs of a function in $\mathcal{C}^k_{ttg}(Q)$, it must be the case that $\ell \leq k$. We further assume that $V(t_1) < V(t_2) < \cdots < V(t_\ell)$. We also define $t_{\ell+1}$ to be an auxiliary task that has $q(t_{\ell+1}) = V(t_{\ell+1}) = \infty$. Next, we obtain weights for the players and thresholds for the tasks. For every set S_j it must be the case that if $v_j = V(t_r)$, then $w(S_j) \geq q(t_r)$, but S_j does not have sufficient weight to complete t_{r+1}. Let $\sigma : [m] \to [\ell]$ be the mapping that, for each sample (S_j, v_j), maps S_j to the task that it completed; i.e. the task t_r for which $v_j = V(t_r)$. This leads to the following linear feasibility problem

$$\begin{aligned} &\text{find: } \mathbf{w} \in \mathbb{R}^n_+, \mathbf{q} \in \mathbb{R}^\ell_+ && && (1)\\ &\text{s.t.: } w(S_j) \geq q(t_{\sigma(j)}) && \forall j \in [m] \\ &\qquad w(S_j) \leq q(t_{\sigma(j)+1}) && \forall j \in [m] \end{aligned}$$

The linear feasibility program (1) has $n + \ell$ variables and $2m$ constraints, and is thus solvable in polynomial time. Moreover, a feasible solution exists; namely, the one that corresponds to the weights in the original TTG. Thus, there is an efficient, consistent algorithm for $\mathcal{C}^k_{ttg}(Q)$. □

Let $\mathcal{C}^k_{ttg}$ be the class of TTGs with k tasks; the following lemma shows that if we take a sufficient number of samples, a game $v \in \mathcal{C}^k_{ttg}$ can be PAC approximated by a game $\bar{v} \in \mathcal{C}^k_{ttg}(Q)$, where Q are the observed values of v.

Lemma 4.6. *Given $m \geq k\frac{1}{\varepsilon}\log\frac{1}{\delta}$ independent samples*

$$(S_1, v(S_1)), \ldots, (S_m, v(S_m))$$

from $v \in \mathcal{C}^k_{ttg}$; let $Q = \bigcup_{j=1}^m \{v(S_j)\}$. The event $\Pr_{S\sim\mathcal{D}}[v(S) \notin Q] < \varepsilon$ occurs with probability at most $1 - \delta$.

Proof. First, recall that for every TTG in $\mathcal{C}^k_{ttg}$, we have k different task values

$$\{V_1, \ldots, V_k\},$$

and any set of observed samples will show some $Q \subseteq \{V_1, \ldots, V_k\}$; thus, there can be at most 2^k observed sets of values from the samples. Let us write $\mathcal{D}^m$ to be the probability distribution from which our samples are taken. Given $Q = \bigcup_{j=1}^m \{v(S_j)\}$, let $Y_\varepsilon(Q)$ be the event that $\Pr_{S\sim\mathcal{D}}[v(S) \notin Q] < \varepsilon$. We need to bound the probability (over samples from $\mathcal{D}^m$) that the event $\neg Y_\varepsilon(Q)$ occurs. Let $\mathcal{S}_m(Q)$ be the set of all sequences sampled from $\mathcal{D}^m$ for which the observed set of values is Q.

First, we note that if $S_1, \ldots, S_m$ generate the values Q, such that $\neg Y_\varepsilon(Q)$ occurs, then $\Pr_{S\sim\mathcal{D}}[v(S) \in Q] \leq 1 - \varepsilon$. Next, note that if $(S_1, \ldots, S_m) \in \mathcal{S}_m(Q)$ then

$v(S_j) \in Q$ for all $j = 1, \ldots, m$. Thus,

$$\Pr[(S_1, \ldots, S_m) \in \mathcal{S}_m(Q)] \leq \prod_{j=1}^{m} \Pr_{S \sim \mathcal{D}}[v(S) \in Q] \leq (1-\varepsilon)^m,$$

which by our choice of m, is at most $\frac{\delta}{2^k}$:

$$\begin{aligned}(1-\varepsilon)^m &\leq \mathrm{e}^{-\varepsilon m} \leq \mathrm{e}^{-\varepsilon k \frac{1}{\varepsilon} \log \frac{1}{\delta}} \\ &= \frac{\delta}{\mathrm{e}^k} < \frac{\delta}{2^k}\end{aligned}$$

Putting it all together,

$$\begin{aligned}\Pr[\neg Y_\varepsilon(Q)] = &\sum_{Q : \neg Y_\varepsilon(Q)} \Pr[(S_1, \ldots, S_m) \in \mathcal{S}_m(Q)] \\ <&2^k \frac{\delta}{2^k} = \delta\end{aligned}$$

which concludes the proof. □

Using the two lemmas, we are now ready to prove that the class of k-TTGs, $\mathcal{C}^k_{ttg}$, is PAC learnable.

Theorem 4.7. *Let $\mathcal{C}^k_{ttg}$ be the class of k-TTGs; then $\mathcal{C}^k_{ttg}$ is PAC learnable.*

Proof Sketch. Let $(S_1, v(S_1)), \ldots, (S_m, v(S_m))$ be our set of samples. According to Lemma 4.6, we can choose m such that with probability $\geq 1 - \frac{\delta}{2}$, $\Pr_{S \sim \mathcal{D}}[v(S) \notin Q] < \frac{\varepsilon}{2}$. We let $\bar{v}$ be the TTG v with the set of tasks reduced to Q; that is $\bar{v}(S) = v(S)$ if $v(S) \in Q$, and is the value of the best task that S can complete whose value is in Q otherwise. Thus, we can pretend that our input is from $\bar{v} \in \mathcal{C}^k_{ttg}(Q)$. According to Lemma 4.5, if m is sufficiently large, then with probability $\geq 1 - \frac{\delta}{2}$ we will output some $v^* \in \mathcal{C}^k_{ttg}(Q)$ such that $\Pr_{S \sim \mathcal{D}}[\bar{v}(S) = v^*(S)] \geq 1 - \frac{\varepsilon}{2}$. Thus, with probability $\geq 1 - \delta$, we have that both $\Pr_{S \sim \mathcal{D}}[\bar{v}(S) = v^*(S)] \geq 1 - \frac{\varepsilon}{2}$ and $\Pr_{S \sim \mathcal{D}}[v(S) = \bar{v}(S)] \geq 1 - \frac{\varepsilon}{2}$. We claim that v^* PAC approximates v. Indeed,

$$\begin{aligned}\Pr_{S \sim \mathcal{D}}[v(S) \neq v^*(S)] = &\Pr_{S \sim \mathcal{D}}[(v(S) \neq v^*(S)) \wedge (v(S) = \bar{v}(S))] \\ &+ \Pr_{S \sim \mathcal{D}}[(v(S) \neq v^*(S)) \wedge (v(S) \neq \bar{v}(S))] \\ &\leq \Pr_{S \sim \mathcal{D}}[v^*(S) \neq \bar{v}(S)] + \Pr_{S \sim \mathcal{D}}[v(S) \neq \bar{v}(S)] < \varepsilon\end{aligned}$$

□

We observe that the output of the algorithm described in Theorem 4.7 is a TTG for which coalition values do not exceed values in the original TTG. However, this does not guarantee that we can obtain a stable outcome using this method, *unless we assume that the value of the original game is known*. Indeed, it is possible that the core of the original game is not empty, whereas the learned game has an empty core.

Finally, even if we are able to PAC learn and output a TTG with a non-empty core, it is not necessarily the case that a core outcome can be computed in polynomial time. This is because computing the core of a TTG is known to be NP-hard [Chalkiadakis *et al.*, 2010], unless weights are given in unary (i.e. the bit precision is polylogarithmic).

Corollary 4.8. *If the value $v(N)$ is known, the class of TTGs with poly-size weights and values is PAC stabilizable.*

4.3 Induced Subgraph Games

An induced subgraph game (ISG) is given by a weighted graph $\Gamma = \langle N, E \rangle$, where for every pair $i, j \in N$, $w_{ij} \in \mathbb{Z}$ denotes the weight of the edge between i and j. We let W be the weighted adjacency matrix of Γ. The value of a coalition $S \subseteq N$ is given by $v(S) = \sum_{i \in S} \sum_{j \in S | j > i} w_{ij}$; i.e. the value of a set of nodes is the weight of their induced subgraph.

Theorem 4.9. *The class of induced subgraph games is efficiently PAC learnable.*

Proof. Let W be the (unknown) weighted adjacency matrix of Γ. Let us write $\mathbf{e}_S$ to be the indicator vector for the set S in $\mathbb{R}^n$. That is, the i-th coordinate of $\mathbf{e}_S$ is 1 if $i \in S$, and is 0 otherwise. We observe that in an ISG, $v(S) = \mathbf{e}_S^T W \mathbf{e}_S$. In other words, learning the coefficients of an ISG is equivalent to learning a linear function with $\mathcal{O}(n^2)$ variables (one per vertex pair), which is known to have polynomial sample complexity [Anthony and Bartlett, 2009].

Now, given observations $(S_1, v_1), \ldots, (S_m, v_m)$, we need to solve a linear system with m constraints (one per sample), and $\mathcal{O}(n^2)$ variables (one per vertex pair, as above), which is solvable in polynomial time.

$$\begin{aligned} &\text{Find:} (w_{i,i'})_{i,i' \in N} && (2) \\ &\text{s.t.} \sum_{i,i' \in S_j} w_{i,i'} = v_j && \forall j = 1, \ldots, m \end{aligned}$$

The output of (2) is guaranteed to be consistent, and since a solution exists (namely, W), we have a straightforward consistent poly-time algorithm, and conclude that the class of ISGs is efficiently PAC learnable. □

It is well known that computing a core outcome for ISGs is NP-hard [Deng and Papadimitriou, 1994], unless all weights are non-negative (in which case the core is never empty). In order to ensure that we find a PAC stable outcome for the latter case, we can slightly modify the solution by searching for a non-negative solution. If a solution exists, we have obtained an outcome that is PAC stable; if not, we drop the non-negativity assumption, but are not guaranteed a poly-time algorithm for finding a core outcome, nor its existence.

4.4 Additional Classes of Cooperative Games

Before we conclude, we present a brief overview of additional results obtained for other classes of cooperative games.

k-WVGs: In weighted voting games (WVGs), each player $i \in N$ has an integer weight w_i; the weight of a coalition $S \subseteq N$ is defined as $w(S) = \sum_{i \in S} w_i$. A coalition is *winning* (has value 1) if $w(S) \geq q$, and has a value of 0 otherwise. Here, q is a given *threshold*, or *quota*. The class of k-vector WVGs is a simple generalization of weighted voting games given by Elkind *et al.* [2009]. A k-vector WVG is given by k WVGs: $\langle \mathbf{w}_1; q_1 \rangle, \ldots, \langle \mathbf{w}_k; q_k \rangle$. A set $S \subseteq N$ is winning if it is winning in every one of the k WVGs.

Learning a weighted voting game is equivalent to learning a separating hyperplane, which is known to be easy [Kearns and Vazirani, 1994]. However, learning k-vector WVGs is equivalent to learning the intersection of k-hyperplanes, which is known to be NP-hard even when $k = 2$ [Alekhnovich *et al.*, 2004; Blum and Rivest, 1992; Klivans *et al.*, 2002]. Thus, k-WVGs are not efficiently PAC learnable, unless P=NP; however, since they are simple and monotone, they are PAC stabilizable according to Theorem 3.3.

Coalitional Skill Games: Coalitional Skill Games (CSGs) [Bachrach and Rosenschein, 2008] are another well-studied class of cooperative games. Here, each player i has a skill-set K_i; additionally, there is a list of tasks $\mathcal{T}$, each with a set of required skills κ_t. Given a set of players $S \subseteq N$, let $K(S)$ be the set of skills that the players in S have. Let $\mathcal{T}(S)$ be the set of tasks $\{t \in \mathcal{T} \mid \kappa_t \subseteq K(S)\}$. The value of the set $\mathcal{T}(S)$ can be determined by various utility models; for example, setting $v(S) = |\mathcal{T}(S)|$, or assuming that there is some subset of tasks $\mathcal{T}^* \subseteq \mathcal{T}$ such that $v(S) = 1$ iff $\mathcal{T}^* \subseteq \mathcal{T}(S)$; the former class of CSGs is known as *conjunctive task skill games (CTSGs)*.

PAC learnability of coalitional skill games is generally computationally hard. This holds even if we make some simplifying assumptions; for example, even if we know the set of tasks and their required skills in advance, or if we know the set of skills each player possesses, but the skills required by tasks are unknown. However, we can show that CTSGs are efficiently PAC learnable if player skills are known.

MC-nets: *Marginal Contribution Nets (MC-nets)* [Ieong and Shoham, 2005] provide compact representation for cooperative games. Briefly, an MC-net is given by a list of rules over the player set N, along with values. A rule is a Boolean formula ϕ_j over N, and a value v_j. For example, $r = x_1 \vee x_2 \vee \neg x_3 \rightarrow 7$ assigns a value of 7 to all coalitions containing players 1 and 2, but not player 3. Given a list of rules, the value of a coalition is the sum of all values of rules that apply to it. PAC learning MC-nets can be reduced to PAC learning of DNF formulas, which is believed to be intractable [Klivans and Servedio, 2001].

5 Discussion

Our work is limited to finding outcomes that are likely to be stable for an unknown function. However, learning approximately stable outcomes is a promising research avenue as well. Such results naturally relate approximately stable outcomes — such as the ε and least core [Peleg and Sudhölter, 2007], or the cost of stability [Bachrach *et al.*, 2009] — with PMAC learning algorithms [Balcan and Harvey, 2011], which seek to

approximate a target function (M stands for "mostly") with high accuracy and confidence.

This work has focused on the core solution concept; however, learning other solution concepts is a natural extension. While some solution concepts, such as the nucleolus or the approximate core variants mentioned above, can be naturally extended to cases where only a subset of the coalitions is observed, it is less obvious how to extend solution concepts such as the Shapley value or Banzhaf power index. These concepts depend on the marginal contribution of player i to coalition S, i.e., $v(S \cup \{i\}) - v(S)$. Under the Shapely value, we are interested in the expected marginal contribution when a permutation of the players is drawn uniformly at random, and i joins previous players in the permutation. According to Banzhaf, S is drawn uniformly at random from all subsets that do not include i. Both solution concepts are easy to approximate if we are allowed to draw coalition values from the appropriate distribution [Bachrach *et al.*, 2010] — this is a good way to circumvent computational complexity when the game is known. It would be interesting to understand what guarantees we obtain for arbitrary distributions.

Acknowledgments: This research is partially supported by NSF grants CCF-1451177, CCF-1422910, and CCF-1101283, CCF-1215883 and IIS-1350598, a Microsoft Research Faculty Fellowship, and Sloan Research Fellowships.

References

M. Alekhnovich, M. Braverman, V. Feldman, A.R. Klivans, and T. Pitassi. Learnability and automatizability. In *FOCS*, pages 621–630, 2004.

M. Anthony and P.L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

Y. Bachrach and J.S. Rosenschein. Coalitional skill games. In *AAMAS*, pages 1023–1030, 2008.

Y. Bachrach and N. Shah. Reliability weighted voting games. In *SAGT*, pages 38–49. Springer, 2013.

Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. Rosenschein. The cost of stability in coalitional games. In *SAGT*, pages 122–134. 2009.

Y. Bachrach, E. Markakis, E. Resnick, A.D. Procaccia, J.S. Rosenschein, and A. Saberi. Approximating power indices: Theoretical and empirical analysis. *JAAMAS*, 20(2):105–122, 2010.

Y. Bachrach, I. Kash, and N. Shah. Agent failures in totally balanced games and convex games. In *WINE*, pages 15–29. Springer, 2012.

Y. Bachrach, R. Meir, M. Feldman, and M. Tennenholtz. Solving cooperative reliability games. *arXiv preprint arXiv:1202.3700*, 2012.

A. Badanidiyuru, S. Dobzinski, H. Fu, R. Kleinberg, N. Nisan, and T. Roughgarden. Sketching valuation functions. In *SODA*, pages 1025–1035, 2012.

M.F. Balcan and N.J.A. Harvey. Learning symmetric non-monotone submodular functions. In *STOC*, pages 793–802, 2011.

M.F. Balcan, F. Constantin, S. Iwata, and L. Wang. Learning valuation functions. In *COLT*, pages 4.1 – 4.24, 2012.

A.L. Blum and R.L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127, 1992.

G. Chalkiadakis and C. Boutilier. Bayesian reinforcement learning for coalition formation under uncertainty. In *AAMAS*, pages 1090–1097, 2004.

G. Chalkiadakis, E. Elkind, E. Markakis, M. Polukarov, and N.R. Jennings. Cooperative games with overlapping coalitions. *JAIR*, 39(1):179–216, 2010.

X. Deng and C.H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.

E. Elkind, L.A. Goldberg, P.W. Goldberg, and M. Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

L.A. Hemaspaandra and M. Ogihara. *The complexity theory companion*. Springer Science & Business Media, 2002.

S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *EC*, pages 193–202. ACM, 2005.

M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

A.R. Klivans and R. Servedio. Learning DNF in time $2^{O(n^{\frac{1}{3}})}$. In *STOC*, pages 258–265. ACM, 2001.

A.R. Klivans, R. O'Donnell, and R.A. Servedio. Learning intersections and thresholds of halfspaces. In *FOCS*, pages 177–186, 2002.

S.M. Lahaie and D.C. Parkes. Applying learning algorithms to preference elicitation. In *EC*, pages 180–188, 2004.

S.M. Lahaie, F. Constantin, and D.C. Parkes. More on the power of demand queries in combinatorial auctions: Learning atomic languages and handling incentives. In *IJCAI*, pages 959–964, 2005.

Y. Li and V. Conitzer. Cooperative game solution concepts that maximize stability under noise. In *AAAI*, 2015.

M. Maschler, E. Solan, and S. Zamir. *Game Theory*. Cambridge Unversity Press, 2013.

B. Peleg and P. Sudhölter. *Introduction to the Theory of Cooperative Games*, volume 34 of *Theory and Decision Library. Series C: Game Theory, Mathematical Programming and Operations Research*. Springer, Berlin, second edition, 2007.

L. Pitt and L.G. Valiant. Computational limitations on learning from examples. *JACM*, 35(4):965–984, 1988.

A.D. Procaccia and J.S. Rosenschein. Learning to identify winning coalitions in the pac model. In *AAMAS*, pages 673–675. ACM, 2006.

A. Shashua. Introduction to machine learning: Class notes 67577. *CoRR*, abs/0904.3664, 2009.

M. Zinkevich, A. Blum, and T. Sandholm. On polynomial-time preference elicitation with value queries. In *EC*, pages 176–185, 2003.