# Automated Design of Voting Rules by Learning from Examples

Ariel D. Procaccia, Aviv Zohar, and Jeffrey S. Rosenschein

**Abstract**

While impossibility results have established that no perfect voting rules exist, efficiently designing a voting rule that satisfies at least a given subset of desiderata remains a difficult task. We argue that such custom-built voting rules can be constructed by learning from examples. Specifically, we consider the learnability of the broad, concisely-representable class of scoring rules. Our main result asserts that this class is efficiently learnable in the PAC model. We also discuss the limitations of our approach, and (along the way) we establish a lemma of independent interest regarding the number of distinct scoring rules.

## 1 Introduction

Voting is a well-studied method of preference aggregation, in terms of its theoretical properties, as well as its computational aspects [3, 2]; various practical, implemented applications exist [9, 8]. In an election, a set of $n$ voters express their preferences over a set of $m$ candidates or alternatives. To be precise, each voter is assumed to reveal linear preferences — a ranking of the candidates. The outcome of the election is determined according to a *voting rule*.

### 1.1 Scoring Rules

The predominant — ubiquitous, even — voting rule in real-life elections is the *Plurality* rule. Under Plurality, each voter awards one point to the candidate it ranks first, i.e., its most preferred alternative. The candidate that accumulated the most points, summed over all voters, wins the election. Another example of a voting rule is the *Veto* rule: each voter "vetoes" a single candidate; the candidate that was vetoed by the fewest voters wins the election. Yet a third example is the *Borda* rule: every voter awards $m-1$ points to its top-ranked candidate, $m-2$ points to its second choice, and so forth — the least preferred candidate is not awarded any points. Once again, the candidate with the most points is elected.

The abovementioned three voting rules all belong to an important family of voting rules known as *scoring rules*. A scoring rule can be expressed by a vector of parameters $\vec{\alpha} = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$, where each $\alpha_l$ is a real number and $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$. Each voter awards $\alpha_1$ points to its most-preferred alternative, $\alpha_2$ to its second-most-preferred alternative, etc. Predictably, the candidate with the most points wins. Under this unified framework, we can express our three rules as:

| | Majority | Robustness | Manipulation | Communication |
|---|---|---|---|---|
| *Plurality* | Yes | $\geq \frac{m-2}{m-1}$ [11] | $\mathcal{P}$ [3] | $\Theta(n \log m)$ [6] |
| *Borda* | No | $\leq \frac{1}{m}$ [11] | $\mathcal{NP}$-complete [3] | $\Theta(nm \log m)$ [6] |
| *Veto* | No | $\geq \frac{m-2}{m-1}$ [11] | $\mathcal{NP}$-complete [2] | ? |

Table 1: Different scoring rules greatly differ in the desiderata they satisfy.

- *Plurality*: $\vec{\alpha} = \langle 1, 0, \ldots, 0 \rangle$.
- *Borda*: $\vec{\alpha} = \langle m-1, m-2, \ldots, 0 \rangle$.
- *Veto*: $\vec{\alpha} = \langle 1, \ldots, 1, 0 \rangle$.

## 1.2 Motivation

Voting rules are often compared on the basis of different criteria, which define potentially desirable properties. We outline below several important criteria, some theoretical, and some computational.

1. *Majority*: If there is a candidate that is most preferred by a majority of voters, does this candidate win the election?
2. *Robustness* [11]: What is the worst-case probability of the outcome of the election *not* changing as a result of a random mistake/fault in the preferences of the voters?
3. *Complexity of Manipulation*: Say a coalition of voters aims to improve its utility from the election by voting untruthfully. How computationally difficult is it to find an optimal vote?
4. *Communication Complexity*: How much communication is required in order to determine the winner of the election?

Impossibility theorems imply that one cannot expect one voting rule to satisfy all desirable criteria simultaneously. However, different voting rules, satisfy different subsets of criteria. In particular, scoring rules greatly differ in this respect. To put it differently, different choices of the parameters of a scoring rule yield significantly different voting rules in terms of their properties. As an example, Table 1 compares Plurality, Borda, and Veto, on the basis of the abovementioned four properties.

## 1.3 Our Approach

So, how would one go about designing a scoring rule with certain properties, configuring the parameters to one's needs? In this paper, we do so by learning from examples. The basic setup is as follows: the designer, or teacher, is presented with different constellations of voters' preferences, drawn according to a fixed distribution. For each such preference profile, the teacher answers

with the winning candidate. For example, if the designer wishes the voting rule to satisfy the majority criterion, and is presented with a profile where a candidate is ranked first by a majority of voters, the designer would answer that this candidate is the winner. More generally, it is possible to consider a setting where properties are represented by tables; for each preference profile, the table designates the set of possible winning candidates (candidates that do not violate the desired property). If a voting rule is to satisfy a given combination of properties, then the winner chosen in every profile is a candidate in the intersection of the different sets of possible winners.

Assuming that there exists a *target* scoring rule that meets all the requirements, we would like to produce a scoring rule that is as "close" as possible. This way, the designer could in principle translate the above cumbersome representation of possible winners using tables, to a concisely-represented voting rule that can be easily understood and computed.

By "close" we mean close with respect to the fixed distribution over preference profiles. More precisely, we would like to construct an algorithm that receives pairs of the form (preferences, winner) drawn according to a fixed distribution $D$ over preferences, and outputs a scoring rule, such that the probability according to $D$ that our scoring rule and the target rule agree is as high as possible. Some readers may have realized that, in fact, we wish to learn scoring rules in the framework of the formal learning model — the PAC (Probably Approximately Correct) model; a concise introduction to this model is given in Section 2.

The dimension of a function class is a combinatorial measure of the richness of the class. The dimension of a class is closely related to the number of examples needed to learn it. We give tight bounds on the dimension of the class of scoring rules: an upper bound of $m$, and a lower bound of $m - 3$, where $m$ is the number of candidates in an election. In addition, we show that, given a set of examples, one can efficiently construct a scoring rule that is consistent with the examples, if one exists. Combined, these results imply that the class of scoring rules is efficiently learnable. In other words, given a combination of properties which is satisfied by some scoring rule, it is possible to construct a "close" scoring rule in polynomial time.

The main weakness of our approach is that there might be cases where there is no scoring rule that satisfies a given combination of properties, although there is a voting rule that does. In this case, there might not exist a scoring rule which is consistent with the given training set. We discuss the limitations of our approach, showing the there are voting rules which cannot be "approximated" by scoring rules. Along the way, we show that the number of distinct scoring rules is at most exponential in the number of voters and candidates (whereas the number of voting rules is double exponential).

### 1.4 Related Work

To the best of our knowledge, we are the first to study automated design of voting rules, and the first to suggest learning as a method of designing social choice mechanisms (although learning is known to be useful in economic settings; PAC learning has very recently been applied to computing utility functions that are rationalizations of given sequences of prices and demands [1]).

Conitzer and Sandholm [4] have studied automated mechanism design, in the more restricted setting where agents have numerical valuations for different alternatives. They propose automatically designing a truthful mechanism for every preference aggregation setting. However, they find that, under two solution concepts, even determining whether there exists a deterministic mechanism that guarantees a certain social welfare is an $\mathcal{NP}$-complete problem. The authors also show that the problem is tractable when designing a randomized mechanism. In more recent work [5], Conitzer and Sandholm put forward an efficient algorithm for designing deterministic mechanisms, which works only in very limited scenarios.

In short, our setting, goals, and methods are completely different — in the general voting context, even framing computational complexity questions is problematic, since the goal cannot be specified with reference to expected social welfare.

### 1.5 Structure of the Paper

In Section 2 we give an introduction to the PAC model. In Section 3, we describe our setting and rigorously prove that the class of scoring rules is efficiently learnable. In Section 4, we discuss the limitations of our approach, and in Section 5, we give our conclusions.

## 2 Preliminaries

In this section we give a very short introduction to the PAC model and the generalized dimension of a function class. A more comprehensive (and slightly more formal) overview of the model, and results concerning the dimension, can be found in [10].

In the PAC model, the learner is attempting to learn a function $f : X \to Y$, which belongs to a class $\mathcal{F}$ of functions from $X$ to $Y$. The learner is given a *training set* — a set of points in $X$, $x_1, x_2, \ldots, x_t$, which are sampled i.i.d. (independently and identically distributed) according to a distribution $D$ over the sample space $X$. $D$ is unknown, but is fixed throughout the learning process. In this paper, we assume the "realizable" case, where a target function $f^*(x)$ exists, and the given training examples are in fact labeled by the target function: $\{(x_k, f^*(x_k))\}_{k=1}^t$. The *error* of a function $f \in \mathcal{F}$ is defined as

$$err(f) = \Pr_{x \sim D}[f(x) \neq f^*(x)]. \tag{1}$$

$\epsilon > 0$ is a parameter given to the learner that defines the *accuracy* of the learning process: we would like to achieve $err(h) \leq \epsilon$. Notice that $err(f^*) = 0$. The learner is also given an *accuracy* parameter $\delta > 0$, that provides an upper bound on the probability that $err(h) > \epsilon$:

$$\Pr[err(h) > \epsilon] < \delta. \tag{2}$$

We now formalize the discussion above:

**Definition 1.**

1. *A* learning algorithm *$L$ is a function from the set of all training examples to $\mathcal{F}$ with the following property: given $\epsilon, \delta \in (0,1)$ there exists an integer $s(\epsilon, \delta)$ — the* sample complexity *— such that for any distribution $D$ on $X$, if $Z$ is a sample of size at least $s$ where the samples are drawn i.i.d. according to $D$, then with probability at least $1 - \delta$ it holds that $err(L(Z)) \leq \epsilon$.*

2. *$L$ is an* efficient *learning algorithm if it always runs in time polynomial in $1/\epsilon$, $1/\delta$, and the size of the representations of the target function, of elements in $X$, and of elements in $Y$.*

3. *A function class $\mathcal{F}$ is* (efficiently) PAC-learnable *if there is an (efficient) learning algorithm for $\mathcal{F}$.*

The sample complexity of a learning algorithm for $\mathcal{F}$ is closely related to a measure of the class's combinatorial richness known as the generalized dimension.

**Definition 2.** Let $\mathcal{F}$ be a class of functions from $X$ to $Y$. We say $\mathcal{F}$ *shatters* $S \subseteq X$ if there exist two functions $g, h \in \mathcal{F}$ such that

1. For all $x \in S$, $g(x) \neq h(x)$.

2. For all $S_1 \subseteq S$, there exists $f \in \mathcal{F}$ such that for all $x \in S_1$, $f(x) = h(x)$, and for all $x \in S \setminus S_1$, $f(x) = g(x)$.

**Definition 3.** Let $\mathcal{F}$ be a class of functions from a set $X$ to a set $Y$. The *generalized dimension* of $\mathcal{F}$, denoted by $D_G(\mathcal{F})$, is the greatest integer $d$ such that there exists a set of cardinality $d$ that is shattered by $\mathcal{F}$.

The generalized dimension of a function provides both upper and lower bounds on the sample complexity of algorithms.

**Theorem 1.** *[10, Theorem 5.1] Let $\mathcal{F}$ be a class of functions from $X$ to $Y$ of generalized dimension $d$. Let $L$ be an algorithm such that, when given a set of $t$ labeled examples $\{(x_k, f^*(x_k))\}_k$ of some $f^* \in \mathcal{F}$, sampled i.i.d. according to some fixed but unknown distribution over the instance space $X$, produces an*

*output $f \in \mathcal{F}$ that is consistent with the training set. Then $L$ is an $(\epsilon, \delta)$-learning algorithm for $\mathcal{F}$ provided that the sample size obeys:*

$$s \geq \frac{1}{\epsilon}\left((\sigma_1 + \sigma_2 + 3)D_G(\mathcal{F}) \ln 2 + \ln\left(\frac{1}{\delta}\right)\right) \tag{3}$$

*where $\sigma_1$ and $\sigma_2$ are the sizes of the representation of elements in $X$ and $Y$, respectively.*

**Theorem 2.** *[10, Theorem 5.2] Let $\mathcal{F}$ be a function class of generalized dimension $d \geq 8$. Then any $(\epsilon, \delta)$-learning algorithm for $\mathcal{F}$, where $\epsilon \leq 1/8$ and $\delta < 1/4$, must use sample size $s \geq \frac{d}{16\epsilon}$.*

# 3 Learning Scoring rules

Before diving in, we introduce some notation. Let $N = \{1, 2, \ldots, n\}$ be the set of voters, and let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of candidates. Let $\mathcal{L}$ be the set of linear preferences[1] over $C$; each voter has preferences $\succ^i \in \mathcal{L}$. We denote the *preference profile*, consisting of the voters' preferences, by $\succ^N = \langle \succ^1, \succ^2, \ldots, \succ^n \rangle$.

Let $\vec{\alpha}$ be a vector of real numbers such that $\alpha_l \geq \alpha_{l+1}$ for all $l = 1, \ldots, m-1$. Let $f_{\vec{\alpha}} : \mathcal{L}^N \to C$ be the scoring rule defined by the vector $\vec{\alpha}$, i.e., each voter awards $\alpha_l$ points to the candidate it ranks in the $l$'th place, and the rule elects the candidate with the most points.

Since several candidates may have maximal scores in an election, we must adopt some method of tie-breaking. Our method works as follows: ties are broken in favor of the candidate that was ranked first by more voters; if several candidates have maximal scores and were ranked first by the same number of voters, the tie is broken in favor of the candidate that was ranked second by more voters; and so on.[2]

Let $\mathcal{S}_m^n$ be the class of scoring rules with $n$ voters and $m$ candidates. Our goal is to learn, in the PAC model, some target function $f_{\vec{\alpha}^*} \in \mathcal{S}_m^n$. To this end, the learner receives a training set $\{(\succ_k^N, f_{\vec{\alpha}^*}(\succ_k^N)\}_k$, where each $\succ_k^N$ is drawn from a fixed distribution over $\mathcal{L}^N$; let $c_{j_k} = f_{\vec{\alpha}^*}(\succ_k^N)$. For the profile $\succ_k^N$, we denote by $\pi_{j,l}^k$ the number of voters that ranked candidate $c_j$ in place $l$. Notice that candidate $c_j$'s score under the preference profile $\succ_k^N$ is $\sum_l \pi_{j,l}^k \alpha_l$.

Our main goal in this section is to prove the following theorem.

**Theorem 3.** *For all $n, m \in \mathbb{N}$, the class $\mathcal{S}_m^n$ is efficiently PAC-learnable.*

By Theorem 1, in order to prove Theorem 3 it is sufficient to validate the following two claims: that there exists an algorithm which, for any training set, runs in time polynomial in the size of the training set and in $n, m$, and outputs

[1] A binary relation which is antisymmetric, transitive, and total.

[2] In case several candidates have maximal scores and identical rankings everywhere, break ties arbitrarily — say, in favor of the candidate with the smallest index.

a scoring rule which is consistent with the training set (assuming one exists); and that the generalized dimension of the class $\mathcal{S}_m^n$ is polynomial in $n$ and $m$.

It is rather straightforward to construct an efficient algorithm that outputs consistent scoring rules. Given a training set, we must choose the parameters of our scoring rule in a way that, for any example, the score of the designated winner is at least as large as the scores of other candidates. Moreover, if ties between the winner and a loser would be broken in favor of the loser, then the winner's score must be strictly higher than the loser's. Our algorithm, given as Algorithm 1, simply formulates all the constraints as linear inequalities, and solves the resulting linear program.

**Algorithm 1** Given a training set, the algorithm returns a scoring rule which is consistent with the given examples, if one exists.

**for** $k \leftarrow 1 \dots t$ **do**
  $C_k \leftarrow \emptyset$
  **for all** $j \neq j_k$ **do** ▷ $c_{j_k}$ is the winner in example $k$
    $\vec{\pi}^{\Delta} \leftarrow \vec{\pi}^k_{j_k} - \vec{\pi}^k_j$
    $l_0 \leftarrow \min\{l : \ \pi^{\Delta}_l \neq 0\}$
    **if** $\pi^{\Delta}_{l_0} < 0$ **then** ▷ Ties are broken in favor of $c_j$
      $C_k \leftarrow C_k \cup \{c_j\}$
    **end if**
  **end for**
**end for**
**return** a feasible solution $\vec{\alpha}$ to the following linear program:

$$\begin{array}{l} \forall k, \ \forall c_j \in C_k, \sum_l \pi^k_{j_k,l}\alpha_l > \sum_l \pi^k_{j,l}\alpha_l \\ \forall k, \ \forall c_j \notin C_k, \sum_l \pi^k_{j_k,l}\alpha_l \geq \sum_l \pi^k_{j,l}\alpha_l \\ \forall l = 1, \dots, m-1 \quad \alpha_l \geq \alpha_{l+1} \\ \forall l, \ \alpha_l \geq 0 \end{array}$$

A linear program can be solved in time that is polynomial in the number of variables and inequalities [12]; it follows that Algorithm 1's running time is polynomial in $n$, $m$, and the size of the training set.

So, it remains to demonstrate that the generalized dimension of $\mathcal{S}_m^n$ is polynomial in $n$ and $m$. The following lemma shows this.

**Lemma 4.** *The generalized dimension of the class $S_m^n$ is at most $m$:*

$$D_G(S_m^n) \leq m.$$

*Proof.* According to Definition 3, we need to show that any set of cardinality $m+1$ cannot be shattered by $\mathcal{F}$. Let $S = \{\succ_k^N\}_{k=1}^{m+1}$ be such a set, and let $h, g$ be the two social choice functions that disagree on all preference profiles in $S$.

We shall construct a subset $S_1 \subseteq S$ such that there is no scoring rule $f_{\vec{\alpha}}$ that agrees with $h$ on $S_1$ and agrees with $g$ on $S \setminus S_1$.

Let us look at the first preference profile from our set, $\succ_1^N$. We shall assume without loss of generality that $h(\succ_1^N) = c_1$, while $g(\succ_1^N) = c_2$, and that in $\succ_1^N$ ties are broken in favor of $c_1$. Let $\vec{\alpha}$ be some parameter vector. If we are to have $h(\succ_1^N) = f_{\vec{\alpha}}(\succ_1^N)$, it must hold that

$$\sum_{l=1}^{m} \pi_{1,l}^1 \cdot \alpha_l \geq \sum_{l=1}^{m} \pi_{2,l}^1 \cdot \alpha_l, \tag{4}$$

whereas if we wanted $f_{\vec{\alpha}}$ to agree with $g$ we would want the opposite:

$$\sum_{l=1}^{m} \pi_{1,l}^1 \cdot \alpha_l < \sum_{l=1}^{m} \pi_{2,l}^1 \cdot \alpha_l \tag{5}$$

More generally, we define, with respect to the profile $\succ_k^N$, the vector $\vec{\pi}_\Delta^k$ as the vector whose $l$'th coordinate is the difference between the number of times the winner under $h$ and the winner under $g$ were ranked in the $l$'th place:[3]

$$\vec{\pi}_\Delta^k = \vec{\pi}_{h(\succ_k)}^k - \vec{\pi}_{g(\succ_k)}^k. \tag{6}$$

Now we can concisely write necessary conditions for $f_{\vec{\alpha}}$ agreeing with $h$ or $g$, respectively, by writing:[4]

$$\vec{\pi}_\Delta^k \cdot \vec{\alpha} \geq 0 \tag{7}$$

$$\vec{\pi}_\Delta^k \cdot \vec{\alpha} \leq 0 \tag{8}$$

Notice that each vector $\vec{\pi}_\Delta^k$ has exactly $m$ coordinates. Since we have $m+1$ such vectors (corresponding to the $m+1$ profiles in $S$), there must be a subset of vectors that is linearly dependent. We can therefore express one of the vectors as a linear combination of the others. W.l.o.g. we assume that the first profile's vector can be written as a combination of the others with parameters $\beta_k$, not all 0:

$$\vec{\pi}_\Delta^1 = \sum_{k=2}^{m+1} \beta_k \cdot \vec{\pi}_\Delta^k \tag{9}$$

Now, we shall construct our subset $S_1$ of preference profiles, on which $f_{\vec{\alpha}}$ agrees with $h$, as follows:

$$S_1 = \{k \in \{2, \ldots, m+1\} : \ \beta_k \geq 0\} \tag{10}$$

[3]There is some abuse of notation; if $h(\succ_k^N) = c_l$ then by $\vec{\pi}_{h(\succ_k)}^k$ we mean $\vec{\pi}_l^k$.

[4]In all profiles except $\succ_1^N$, we are indifferent to the direction in which ties are broken.

Suppose, by way of contradiction, that $f_{\vec{\alpha}}$ agrees with $h$ on $\succ_k^N$ for $k \in S_1$, and with $g$ on the rest. We shall examine the value of $\vec{\pi}_\Delta^1 \cdot \vec{\alpha}$:

$$\vec{\pi}_\Delta^1 \cdot \vec{\alpha} = \sum_{k=2}^{m+1} \beta_k \cdot \vec{\pi}_\Delta^k \cdot \vec{\alpha} = \sum_{k \in S_1} \beta_k \cdot \vec{\pi}_\Delta^k \cdot \vec{\alpha} + \sum_{k \notin S_1 \cup \{1\}} \beta_k \cdot \vec{\pi}_\Delta^k \cdot \vec{\alpha} \geq 0 \qquad (11)$$

The last inequality is due to the construction of $S_1$ — whenever $\beta_k$ is negative, the sign of $\vec{\pi}_\Delta^k \cdot \vec{\alpha}$ is non-positive ($f_{\vec{\alpha}}$ agrees with $g$), and whenever $\beta_k$ is positive, the sign of $\vec{\pi}_\Delta^k \cdot \vec{\alpha}$ is non-negative (agreement with $h$).

Therefore, by equation (5), we have that $f(\succ_1^N) \neq c_2 = g(\succ_1^N)$. However, it holds that $1 \notin S_1$, and we assumed that $f_{\vec{\alpha}}$ agrees with $g$ outside $S_1$ — this is a contradiction. □

Theorem 3 is thus proven. The upper bound on the generalized dimension of $S_m^n$ is quite tight: in the next subsection we show a lower bound of $m - 3$.

## 3.1 Lower Bound for the Generalized Dimension of $\mathcal{S}_m^n$

Theorem 2 implies that a lower bound on the generalized dimension of a function class is directly connected to the complexity of learning it. In particular, a tight bound on the dimension gives us an almost exact idea of the number of examples required to learn a scoring rule. Therefore, we wish to bound $D_G(\mathcal{S}_m^n)$ from below as well.

**Theorem 5.** *For all $n \geq 4$, $m \geq 4$, $D_G(\mathcal{S}_m^n) \geq m - 3$.*

*Proof.* We shall produce an example set of size $m - 3$ which is shattered by $\mathcal{S}_m^n$. Define a preference profile $\succ_l^N$, for $l = 3, \ldots, m-1$, as follows. For all $l$, the voters $1, \ldots, n-1$ rank candidate $c_j$ in place in place $j$, i.e., they vote $c_1 \succ_l^i c_2 \succ_l^i \cdots \succ_l^i c_m$. The preferences $\succ_l^n$ (the preferences of voter $n$ in profile $\succ_l^N$) are defined as follows: candidate 2 is ranked in place $l$, candidate 1 is ranked in place $l+1$; the other candidates are ranked arbitrarily by voter $n$. For example, if $m = 5$, $n = 6$, the preference profile $\succ_3^N$ is:

| $\succ_3^1$ | $\succ_3^2$ | $\succ_3^3$ | $\succ_3^4$ | $\succ_3^5$ | $\succ_3^6$ |
|---|---|---|---|---|---|
| $c_1$ | $c_1$ | $c_1$ | $c_1$ | $c_1$ | $c_3$ |
| $c_2$ | $c_2$ | $c_2$ | $c_2$ | $c_2$ | $c_4$ |
| $c_3$ | $c_3$ | $c_3$ | $c_3$ | $c_3$ | $c_2$ |
| $c_4$ | $c_4$ | $c_4$ | $c_4$ | $c_4$ | $c_1$ |
| $c_5$ | $c_5$ | $c_5$ | $c_5$ | $c_5$ | $c_5$ |

**Lemma 6.** *For any scoring rule $f_{\vec{\alpha}}$ with $\alpha_1 = \alpha_2 \geq 2\alpha_3$ it holds that:*

$$f_{\vec{\alpha}}(\succ_l^N) = \begin{cases} c_1 & \alpha_l = \alpha_{l+1} \\ c_2 & \alpha_l > \alpha_{l+1} \end{cases}$$

*Proof.* We shall first verify that $c_2$ has maximal score. $c_2$'s score is at least $(n-1)\alpha_2 = (n-1)\alpha_1$. Let $j \geq 3$; $c_j$'s score is at most $(n-1)\alpha_3 + \alpha_1$. Thus, the difference is at least $(n-1)(\alpha_1 - \alpha_3) - \alpha_1$. Since $\alpha_1 = \alpha_2 \geq 2\alpha_3$, this is at least $(n-1)(\alpha_1/2) - \alpha_1 > 0$, where the last inequality holds for $n \geq 4$.

Now, under preference profile $\succ_l^N$, $c_1$'s score is $(n-1)\alpha_1 + \alpha_{l+1}$ and $c_2$'s score is $(n-1)\alpha_1 + \alpha_l$. If $\alpha_l = \alpha_{l+1}$, the two candidates have identical scores, but $c_1$ was ranked first by more voters (in fact, by $n-1$ voters), and thus the winner is $c_1$. If $\alpha_l > \alpha_{l+1}$, then $c_2$'s score is strictly higher — hence in this case $c_2$ is the winner. □

Armed with Lemma 6, we prove that the set $\{\succ_l^N\}_{l=3}^{m-1}$ is shattered by $\mathcal{S}_m^n$. Let $\vec{\alpha}^1$ such that $\alpha_1^1 = \alpha_2^1 \geq 2\alpha_3^1 = \alpha_4^1 = \cdots = \alpha_m^1$, and $\vec{\alpha}^2$ such that $\alpha_1^1 = \alpha_2^1 \geq 2\alpha_3^1 > \alpha_4^1 > \cdots > \alpha_m^1$. By the lemma, for all $l = 3, \ldots, m-1$, $f_{\vec{\alpha}^1}(\succ_l^N) = c_1$, and $f_{\vec{\alpha}^2}(\succ_l^N) = c_2$.

Let $T \subseteq \{3, 4, \ldots, m-1\}$. We must show that there exists $\vec{\alpha}$ such that $f_{\vec{\alpha}}(\succ_l^N) = c_1$ for all $l \in T$, and $f_{\vec{\alpha}}(\succ_l^N) = c_2$ for all $l \notin T$. Indeed, configure the parameters such that $\alpha_1 = \alpha_2 > 2\alpha_3$, and $\alpha_l = \alpha_{l+1}$ iff $l \in T$. The result follows directly from Lemma 6. □

# 4 Limitations

Heretofore, we have concentrated on trying to learn scoring rules. In particular, we have assumed that there is a scoring rule that is consistent with given training sets. We have motivated our attention to this specific family of rules by demonstrating that it is possible to obtain a variety of properties by adjusting the parameters that define scoring rules.

In this section, we push the envelope by asking the following question. Given examples that are consistent with some general voting rule, is it possible to learn a scoring rule that is "close" to the target rule? The natural definition of distance, in this case, would seem to be the fraction of preference profiles on which the two rules disagree.

**Definition 4.** A voting rule $f : \mathcal{L}^N \to C$ is a *c-approximation* of a voting rule $g$ iff $f$ and $g$ agree on a $c$-fraction of the possible preference profiles:

$$|\{\succ^N \in \mathcal{L}^N : \ f(\succ^N) = g(\succ^N)\}| \geq c \cdot (m!)^n.$$

In other words, the question is: given a training set $\{(\succ_k^N, f(\succ_j^N)\}_k$, where $f : \mathcal{L}^N \to C$ is some voting rule, how hard is it to learn a scoring rule that $c$-approximates $f$, for $c$ that is close to 1?

It turns out that the answer is: it is impossible. Indeed, there are voting rules that disagree with any scoring rule on half of all preference profiles; if the target rule $f$ is such a rule, it is impossible to find, and of course impossible to learn, a scoring rule that is "close" to $f$.

**Theorem 7.** *Let $\epsilon > 0$. For large enough values of $n$ and $m$, there is a voting rule $F : \mathcal{L}^n \to \{c_1, \ldots, c_m\}$ such that no scoring rule in $\mathcal{S}_m^n$ is a $(1/2 + \epsilon)$-approximation of $F$.*

In order to prove the theorem, we require the following lemma, which may be of independent interest.

**Lemma 8.** *There exists a polynomial $p(n, m)$ such that for all $n, m \in \mathbb{N}$, $|\mathcal{S}_m^n| \leq 2^{p(n,m)}$.*

*Proof.* It is true that there is an infinite number of ways to choose the vector $\vec{\alpha}$ that defines a scoring rule. Nevertheless, what we are really interested in is the number of *distinct* voting rules. For instance, if $\vec{\alpha}^1 = 2\vec{\alpha}^2$, then $f_{\vec{\alpha}^1} \equiv f_{\vec{\alpha}^2}$, i.e., the two vectors define the same voting rule.

It is clear that two scoring rules $f_{\vec{\alpha}^1}$ and $f_{\vec{\alpha}^2}$ are distinct only if the following condition holds: there exist two candidates $c_{j_1}, c_{j_2} \in C$, and a preference profile $\succ^N$, such that $f_{\vec{\alpha}^1}(\succ^N) = c_{j_1}$ and $f_{\vec{\alpha}^2}(\succ^N) = c_{j_2}$. This holds only if there exist two candidates $c_{j_1}$ and $c_{j_2}$ and a preference profile $\succ^N$ such that under $\alpha^1$, $c_{j_1}$'s score is strictly greater than $c_{j_2}$'s, and under $\alpha_2$, either $c_{j_2}$'s score is greater or the two candidates are tied, and the tie is broken in favor of $c_{j_2}$.

Now, assume $\succ^N$ induces rankings $\vec{\pi}_{j_1}$ and $\vec{\pi}_{j_2}$. The conditions above can be written as

$$\sum_l \pi_{j_1,l}\alpha_l^1 > \sum_l \pi_{j_2,l}\alpha_l^1, \tag{12}$$

$$\sum_l \pi_{j_1,l}\alpha_l^2 \leq \sum_l \pi_{j_2,l}\alpha_l^2, \tag{13}$$

where the inequality is an equality only if ties are broken in favor of $c_{j_2}$, i.e., if $l_0 = \min\{l : \pi_{j_1,l} \neq \pi_{j_2,l}\}$, then $\pi_{j_1,l} < \pi_{j_2,l}$.[5]

Let $\vec{\pi}_\Delta = \vec{\pi}_{j_1} - \vec{\pi}_{j_2}$. As in the proof of Lemma 4, equations (12) and (13) can be concisely rewritten as

$$\vec{\pi}_\Delta \cdot \vec{\alpha}^1 > 0 \geq \vec{\pi}_\Delta \cdot \vec{\alpha}^2, \tag{14}$$

where the inequality is an equality only if the first nonzero position in $\vec{\pi}_\Delta$ is negative.

In order to continue, we opt to reinterpret the above discussion geometrically. Each point in $\mathbb{R}^m$ corresponds to a possible choice of parameters $\vec{\alpha}$. Now, each possible choice of $\vec{\pi}_\Delta$ is the normal to a hyperplane. These hyperplanes partition the space into cells: the vectors in the interior of each cell agree on the signs of dot products with all vectors $\vec{\pi}_\Delta$. More formally, if $\vec{\alpha}_1$ and $\vec{\alpha}_2$ are two points in the interior of a cell, then for any vector $\vec{\pi}_\Delta$, $\vec{\pi}_\Delta \cdot \vec{\alpha}^1 > 0 \Leftrightarrow \vec{\pi}_\Delta \cdot \vec{\alpha}^2 > 0$. By equation (14), this implies that any two scoring rules $f_{\vec{\alpha}^1}$ and $f_{\vec{\alpha}^2}$, where $\vec{\alpha}^1$ and $\vec{\alpha}^2$ are in the interior of the same cell, are identical.

[5] W.l.o.g. we disregard the case where $\vec{\pi}_{j_1} = \vec{\pi}_{j_2}$; the reader can verify that taking this case into account multiplies the final result by an exponential factor at most.

What about points residing in the intersection of several cells? These vectors always agree with the vectors in one of the cells, as ties are broken according to rankings induced by the preference profile, i.e., according to the parameters that define our hyperplanes. Therefore, the points in the intersection can be conceptually annexed to one of the cells.

So, we have reached the conclusion that the number of distinct scoring rules is at most the number of cells. Hence, it is enough to bound the number of cells; we claim this number is exponential in $n$ and $m$. Indeed, each $\vec{\pi}_\Delta$ is an $m$-vector, in which every coordinate is an integer in the set $\{-n, -n+1, \ldots, n-1, n\}$. It follows that there are at most $(2n+1)^m$ possible hyperplanes. It is known [7] that given $k$ hyperplanes in $d$-dimensional space, the number of cells is at most $O(k^d)$. In our case, $k \leq (2n+1)^m$ and $d = m$, so we have obtained a bound of:

$$((2n+1)^m)^m \leq (3n)^{m^2} = \left(2^{\log 3n}\right)^{m^2} = 2^{m^2 \log 3n}. \tag{15}$$

□

*Proof of Theorem 7.* We will surround each scoring rule $f_{\vec{\alpha}} \in \mathcal{S}_m^n$ with a "ball" $B(\vec{\alpha})$, which contains all the voting rules for which $f_{\vec{\alpha}}$ is a $(1/2+\epsilon)$-approximation. We will then show that the union of all these balls does not cover the entire set of voting rules. This implies that there is a voting rule for which no scoring rule is a $(1/2+\epsilon)$-approximation.

For a given $\vec{\alpha}$, what is the size of $B(\vec{\alpha})$? As there are $(m!)^n$ possible preference profiles, the ball contains rules that do not agree with $f_{\vec{\alpha}}$ on at most $(1/2-\epsilon)(m!)^n$ preference profiles. For a profile on which there is disagreement, there are $m$ options to set the image under the disagreeing rule.[6] Therefore,

$$|B(\vec{\alpha})| \leq \binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{(1/2-\epsilon)(m!)^n}. \tag{16}$$

How large is this expression? Let $B'(\vec{\alpha})$ be the set of all voting rules that disagree with $f_{\vec{\alpha}}$ on *exactly* $(1/2+\epsilon)(m!)^n$ preference profiles. It holds that

$$\begin{aligned} |B'(\vec{\alpha})| &= \binom{(m!)^n}{(1/2+\epsilon)(m!)^n} (m-1)^{(1/2+\epsilon)(m!)^n} \\ &= \binom{(m!)^n}{(1/2-\epsilon)(m!)^n} ((m-1)^{1+2\epsilon})^{1/2(m!)^n} \\ &\geq \binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{1/2(m!)^n}, \end{aligned} \tag{17}$$

where the last inequality holds for a large enough $m$. But since the total number of voting rules, $m^{(m!)^n}$, is greater than the number of rules in $B'(\vec{\alpha})$, we have:

$$\frac{m^{(m!)^n}}{B(\vec{\alpha})} \geq \frac{B'(\vec{\alpha})}{B(\vec{\alpha})} \geq \frac{\binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{1/2(m!)^n}}{\binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{(1/2-\epsilon)(m!)^n}} = m^{\epsilon(m!)^n}. \tag{18}$$

[6] This way, we also take into account voting rules that agree with $f_{\vec{\alpha}}$ on more than $(1/2+\epsilon)(m!)^n$ profiles.

Therefore

$$B(\vec{\alpha}) \leq \frac{m^{(m!)^n}}{m^{\epsilon(m!)^n}} = m^{(1-\epsilon)(m!)^n}. \tag{19}$$

If the union of balls is to cover the entire set of voting rules, we must have $|\mathcal{S}_m^n| \cdot m^{(1-\epsilon)(m!)^n} \geq m^{(m!)^n}$; equivalently, it must hold that $|\mathcal{S}_m^n| \geq m^{\epsilon(m!)^n}$. However, Lemma 8 implies that $|\mathcal{S}_m^n|$ is exponential in $n$ and $m$, so for large enough values of $n$ and $m$, the above condition does not hold. □

# 5 Conclusions

We have shown that the class of scoring rules is efficiently learnable in the PAC model. We have argued that, given properties the designer would like a voting rule to satisfy, learning from examples allows it to efficiently (albeit approximately) construct such a rule, if indeed one exists. Our basic assumption was that the designer can designate winning candidates in given preference profiles, by consulting some representation of the properties. So, the designer essentially translates a cumbersome representation of properties to a concisely represented voting rule which is easy to understand and apply.

We demonstrated that voting rules can capture a wide variety of properties. However, in Section 4 we explored the limitations of our approach, and showed that many voting rules cannot be approximated using scoring rules. This suggests that for some combinations of properties, there is no scoring rule that is close to satisfying all properties, whereas in general such a voting rule exists. On the other hand, we may have asked for too much. We did not attempt to characterize any of the disagreeing voting rules, and in practice they may be very bizarre. For example, consider the rule that sets the candidate that was most often ranked last as the winner. The abovementioned results raise two important questions, which we intend to investigate in the future:

1. Is there a class of voting rules that is significantly broader than the class of scoring rules, such that any voting rule in the former class can be approximated by a scoring rule?

2. Is there a class of voting rules that is significantly broader than the class of scoring rules, as well as efficiently learnable and concisely representable?

If the answer to one of the questions is "yes", we would be able to circumvent some of the alleged limitations of our approach.

# References

[1] E. Beigman and R. Vohra. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 36–42, 2006.

[2] V. Conitzer, J. Lang, and T. Sandholm. How many candidates are needed to make elections hard to manipulate? In *Proceedings of the International Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 201–214, 2003.

[3] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence*, pages 314–319, 2002.

[4] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence*, pages 103–110, 2002.

[5] V. Conitzer and T. Sandholm. An algorithm for automatically designing deterministic mechanisms without payments. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 128–135, 2004.

[6] V. Conitzer and T. Sandholm. Communication complexity of common voting rules. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 78–87, 2005.

[7] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1987.

[8] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: the anatomy of a recommender system. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 434–435, 1999.

[9] T. Haynes, S. Sen, N. Arora, and R. Nadella. An automated meeting scheduling system that utilizes user preferences. In *Proceedings of the First International Conference on Autonomous Agents*, pages 308–315, 1997.

[10] B. K. Natarajan. *Machine Learning: A Theoretical Approcach*. Morgan Kaufmann, 1991.

[11] A. D. Procaccia, J. S. Rosenschein, and G. A. Kaminka. On the robustness of preference aggregation in noisy environments. In *Proceedings of the First International Workshop on Computational Social Choice*, 2006.

[12] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2nd edition, 2001.

Ariel D. Procaccia, Aviv Zohar, Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
Givat Ram, Jerusalem 91904, Israel
Email: {arielpro,avivz,jeff}@cs.huji.ac.il