Social Choice Theory and Machine Learning
Lecture 3

Eric Pacuit, University of Maryland

August 7, 2024

Plan for today

- ✓ A brief introduction to social choice theory
- ✓ A survey of voting methods
- ✓ Characterizing voting methods
- ✓ Splitting cycles and breaking ties
- Stable Voting
- Preferential Voting Tools
- Learning voting rules

Stable Voting

Our proposed Voting Method is Stable Voting, defined recursively as follows (where ‘undefeated’ is defined according to Split Cycle):

Stable Voting

Our proposed Voting Method is Stable Voting, defined recursively as follows (where ‘undefeated’ is defined according to Split Cycle):

- If there is only one undefeated candidate a, then a wins.

Stable Voting

Our proposed Voting Method is Stable Voting, defined recursively as follows (where ‘undefeated’ is defined according to Split Cycle):

- If there is only one undefeated candidate a, then a wins.
- Otherwise list all head-to-head matches a vs. b, where a is undefeated, in order from the largest to the smallest margin of a vs. b.

Stable Voting

Our proposed Voting Method is Stable Voting, defined recursively as follows (where 'undefeated' is defined according to Split Cycle):

- If there is only one undefeated candidate $\boldsymbol{a}$, then $\boldsymbol{a}$ wins.
- Otherwise list all head-to-head matches $\boldsymbol{a}$ vs. $\boldsymbol{b}$, where $\boldsymbol{a}$ is undefeated, in order from the largest to the smallest margin of $\boldsymbol{a}$ vs. $\boldsymbol{b}$.

  Find the first match such that $\boldsymbol{a}$ wins according to Stable Voting *after $\boldsymbol{b}$ is removed from all ballots*;

Stable Voting

Our proposed Voting Method is Stable Voting, defined recursively as follows (where 'undefeated' is defined according to Split Cycle):

- If there is only one undefeated candidate a, then a wins.
- Otherwise list all head-to-head matches a vs. b, where a is undefeated, in order from the largest to the smallest margin of a vs. b.

  Find the first match such that a wins according to Stable Voting *after b is removed from all ballots*; this a is the winner for the original set of ballots.

Stable Voting

Our proposed Voting Method is Stable Voting, defined recursively as follows (where ‘undefeated’ is defined according to Split Cycle):

- If there is only one undefeated candidate a, then a wins.
- Otherwise list all head-to-head matches a vs. b, where a is undefeated, in order from the largest to the smallest margin of a vs. b.

  Find the first match such that a wins according to Stable Voting *after b is removed from all ballots*; this a is the winner for the original set of ballots.

W. Holliday and E. Pacuit. *Stable Voting*. Constitutional Political Economy, 2023.

Simple Stable Voting

Simple Stable Voting is defined just like Stable Voting except that we list all head-to-head matches a vs. b, rather than only those where a is undefeated:

Simple Stable Voting

Simple Stable Voting is defined just like Stable Voting except that we list all head-to-head matches a vs. b, rather than only those where a is undefeated:

- If only one candidate a appears on all ballots, then a wins.

Simple Stable Voting

Simple Stable Voting is defined just like Stable Voting except that we list all head-to-head matches a vs. b, rather than only those where a is undefeated:

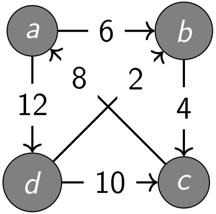
- If only one candidate a appears on all ballots, then a wins.
- Otherwise list all head-to-head matches a vs. b~~, where a is undefeated,~~ in order from the largest to the smallest margin of a vs. b.

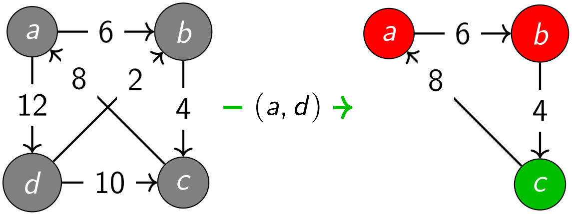
Simple Stable Voting

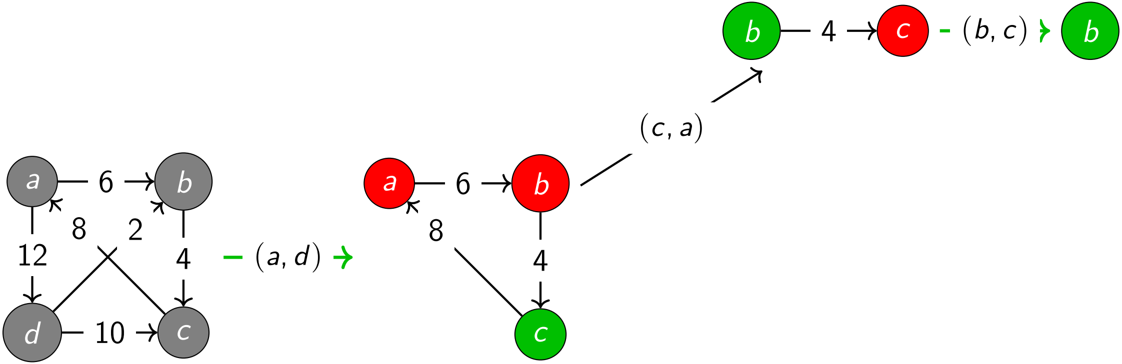
Simple Stable Voting is defined just like Stable Voting except that we list all head-to-head matches a vs. b, rather than only those where a is undefeated:

- If only one candidate a appears on all ballots, then a wins.
- Otherwise list all head-to-head matches a vs. b~~, where a is undefeated,~~ in order from the largest to the smallest margin of a vs. b.

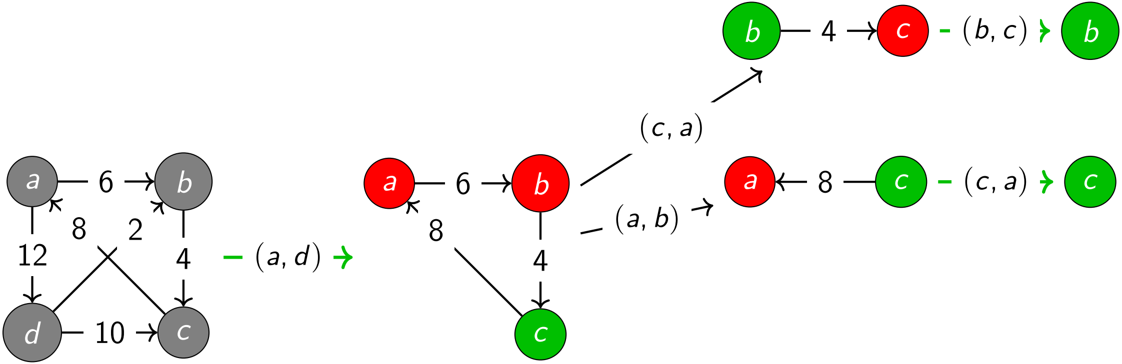
  Find the first match such that a wins according to Simple SV *after b is removed from all ballots*; this a is the winner for the original set of ballots.


a
b
c
d
6
12
8
2
4
10

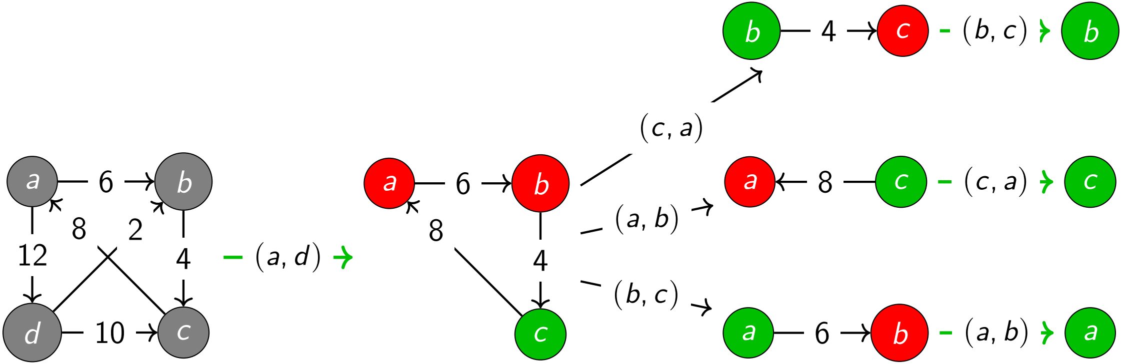

a
6
b
8
2
12
4
d
10
c
− (a, d)
a
6
b
8
4
c



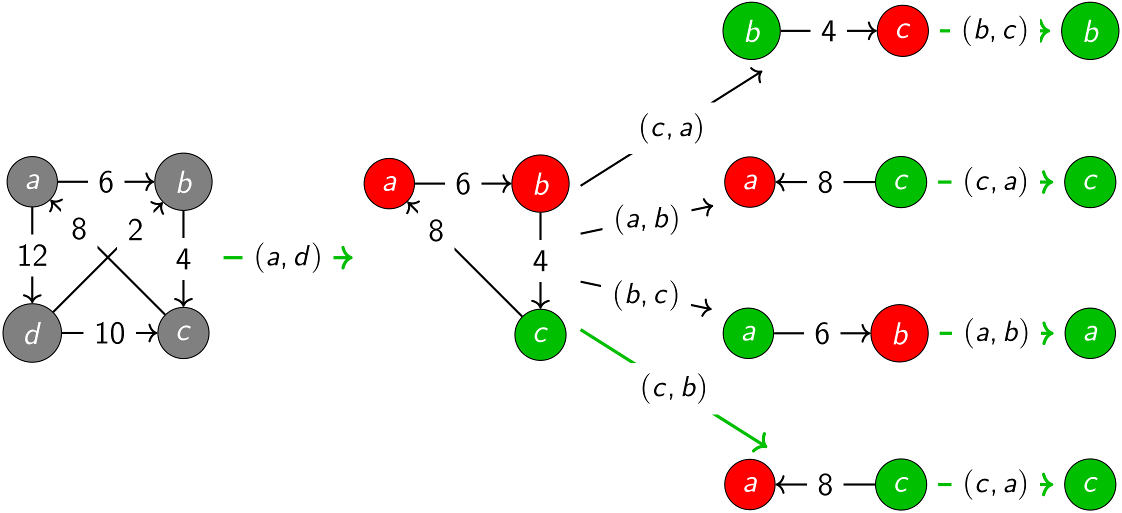
a
6
b
8
2
12
4
d
10
c
(a, d)
a
6
b
8
4
c
(c, a)
b
4
c
(b, c)
b

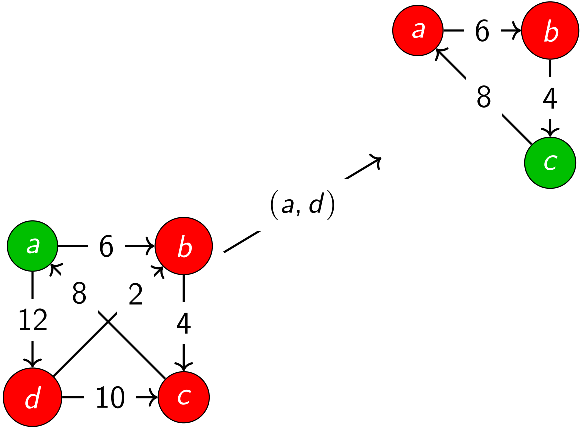


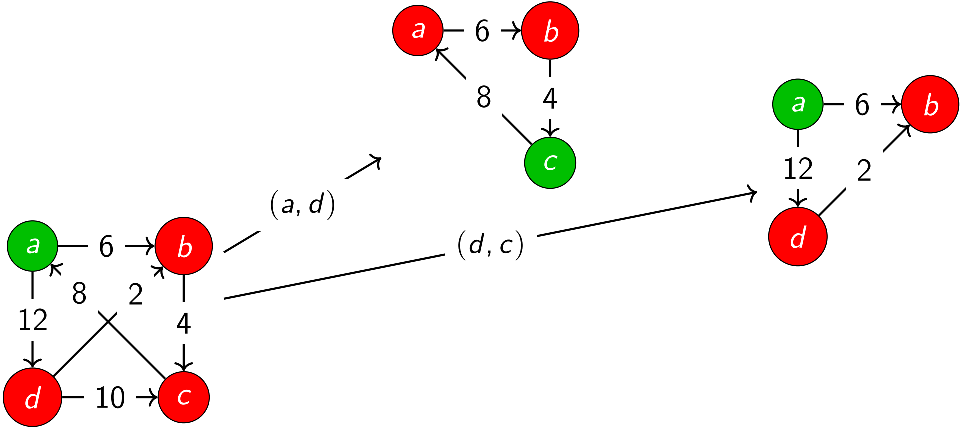
a
b
c
d
6
12
8
2
4
10
(a, d)
(c, a)
(a, b)
(b, c)



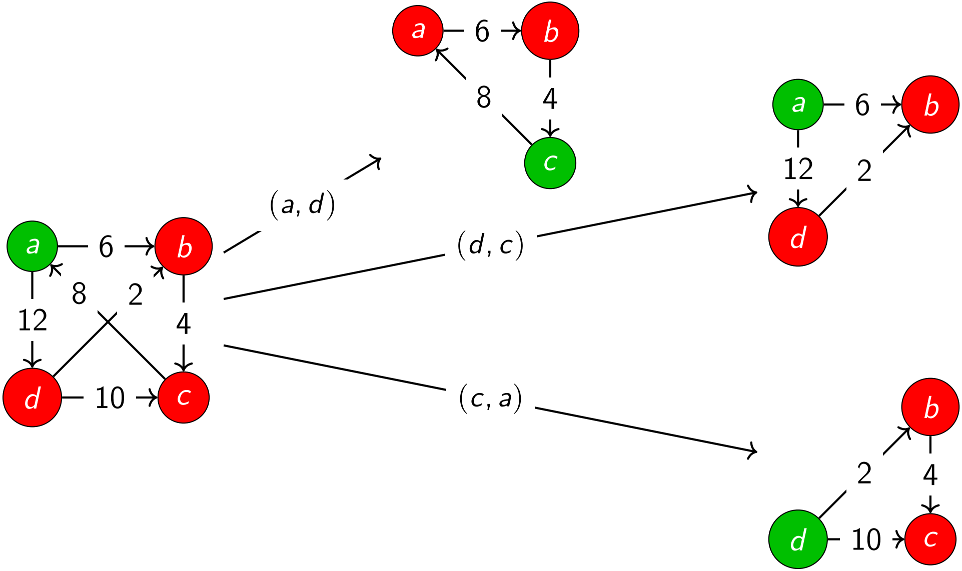
a
b
c
d
6
12
8
2
4
10
(a, d)
(c, a)
(a, b)
(b, c)


a
b
c
d
6
8
2
12
4
10
(a, d)
(c, a)
(a, b)
(b, c)
(c, b)
(c, a)
(a, b)
(b, c)

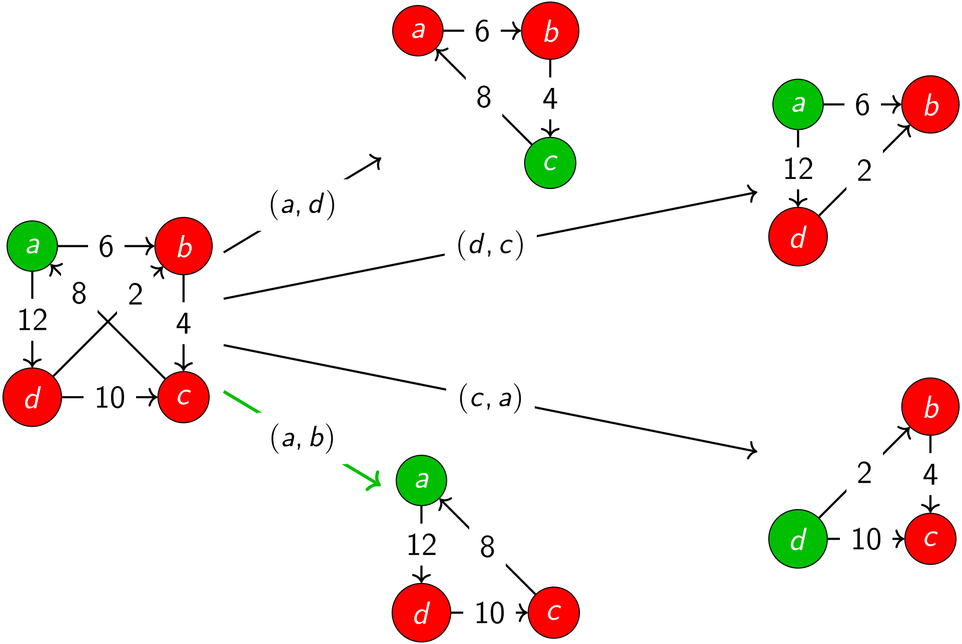

a
6
b
8
4
c
(a, d)
a
6
b
12
8
2
4
d
10
c



a
b
c
d
6
12
8
2
4
10
(a, d)
(d, c)



a
6
b
12
8
2
4
d
10
c
(a, d)
(d, c)
(c, a)



a
b
c
d
6
8
2
12
4
10
(a, d)
(d, c)
(c, a)
(a, b)

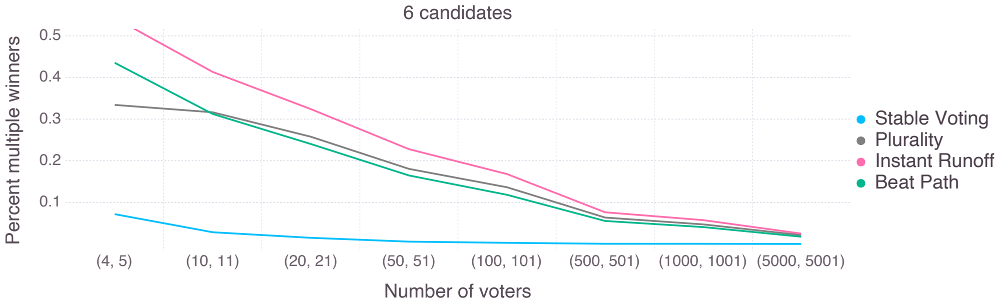
Benefits of Stable Voting

Stable Voting satisfies **Stability for Winners with Tiebreaking** and **Quasi-resoluteness** (and hence **Asymptotic Resolvability**).

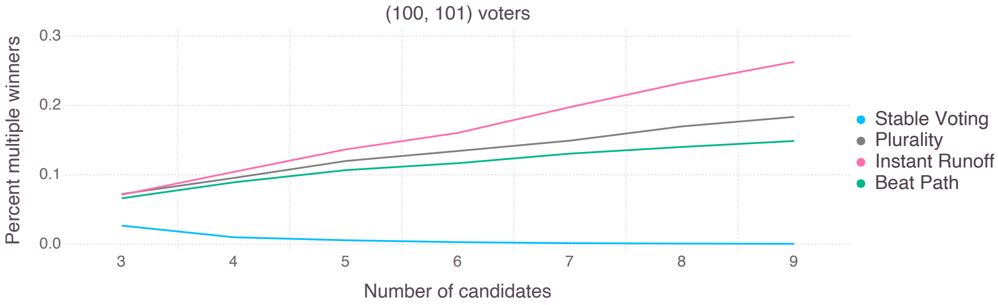
Benefits of Stable Voting

Stable Voting satisfies **Stability for Winners with Tiebreaking** and **Quasi-resoluteness** (and hence **Asymptotic Resolvability**).

In fact, SV has a remarkable ability to avoid ties even in elections with small numbers of voters that can produce tied margins.



6 candidates
Percent multiple winners
0.5
0.4
0.3
0.2
0.1
(4, 5)
(10, 11)
(20, 21)
(50, 51)
(100, 101)
(500, 501)
(1000, 1001)
(5000, 5001)
Number of voters
Stable Voting
Plurality
Instant Runoff
Beat Path



(100, 101) voters
Percent multiple winners
0.3
0.2
0.1
0.0
3
4
5
6
7
8
9
Number of candidates
Stable Voting
Plurality
Instant Runoff
Beat Path

Costs of Stable Voting

For truth in advertising, there are some costs of Stable Voting:

Costs of Stable Voting

For truth in advertising, there are some costs of Stable Voting:

1. Computing the SV winners using our current recursive implementation can be computationally expensive above 20 candidates.

Costs of Stable Voting

For truth in advertising, there are some costs of Stable Voting:

1. Computing the SV winners using our current recursive implementation can be computationally expensive above 20 candidates.
2. There are some violations—in an extremely small fraction of profiles—of voting criteria satisfied by some other voting methods, such as *monotonicity*.

Costs of Stable Voting

For truth in advertising, there are some costs of Stable Voting:

1. Computing the SV winners using our current recursive implementation can be computationally expensive above 20 candidates.
2. There are some violations—in an extremely small fraction of profiles—of voting criteria satisfied by some other voting methods, such as *monotonicity*.

Re 1, we can handle larger profiles that are uniquely weighted with up to 20 candidates in the "Smith set." This covers many voting contexts.

Costs of Stable Voting

For truth in advertising, there are some costs of Stable Voting:

1. Computing the SV winners using our current recursive implementation can be computationally expensive above 20 candidates.
2. There are some violations—in an extremely small fraction of profiles—of voting criteria satisfied by some other voting methods, such as *monotonicity*.

Re 1, we can handle larger profiles that are uniquely weighted with up to 20 candidates in the "Smith set." This covers many voting contexts.

Re 2, the frequency with which Stable Voting violates monotonicity is minuscule compared to the frequency for Instant Runoff (in use in the Bay Area and NYC).

StableVoting.org

Stable Voting Create Poll About Contact

Stable Voting is a free and easy way to make a group decision by voting.

Create a poll and send the generated link to your voters.

Voters rank the candidates.

View the winner and explanation of results.

StableVoting.org

Stable Voting has a remarkably low tie frequency, making it very useful in elections with even small numbers of voters.

Over 200 real elections have already been run on StableVoting.org.

People have voted on all kinds of issues:

- electing leaders and officials, such as presidents of organizations, boards of directors, union representatives;
- choosing names for children, pets, groups, etc.;
- planning social events and gatherings, like trips, parties, and outings;
- soliciting entertainment preferences, about books, TV shows, and movies;
- deciding miscellaneous organizational matters, such as meeting times, rules, and procedures

Please try the website and let us know what you think!

Using axioms to evaluate voting methods

Suppose that a voting method F violates an axiom while the voting method G satisfies the axiom.

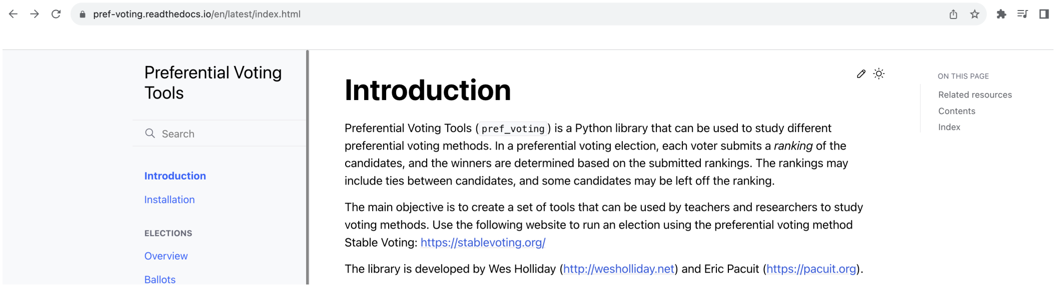
Using axioms to evaluate voting methods

Suppose that a voting method F violates an axiom while the voting method G satisfies the axiom.

- How likely is it that F violates the axiom (with respect to some probability model generating elections)?
- How likely is it that F violates the axiom conditional on F and G selecting different winners (with respect to some probability model generating elections)?

Preferential Voting Tools

More than 70 voting methods (including voting methods for different ballot types) are implemented in our Preferential Voting Tools Python package (`https://pref-voting.readthedocs.io/`)



Brief Introduction to the Preferential Voting Tools

Other ways to evaluate voting methods

- Utility: Given the utilities of the voters, which voting method comes as close as possible to maximizing *social utility*?
- Epistemic: Assuming that the ballots are noisy signals from the voters about some "correct" alternative (or ranking), which voting method is more likely to select the correct alternative (or ranking)?
- Computational: What is the complexity of finding the winner? What information from the voters is needed to compute the winner?
- Strategic: To what extent does the voting method incentivize *strategic* voting? Or strategic agenda setting?

Course Plan

- ✓ introduction to mathematical analysis of voting methods, voting paradoxes;
- ? probabilistic voting methods (skipped for now);
- ✓ quantitative analysis of voting methods (e.g., Condorcet efficiency);
- ▶ learning voting rules (PAC-learning, MLPs, other approaches);
- ▶ using modern deep learning techniques to generate synthetic election data;
- ▶ strategic voting, learning to successfully manipulate voting rules based on limited information about how the other voters will vote using neural networks (multi-layer perceptrons);
- ▶ RLHF (reinforcement learning with human feedback) and social choice;
- ▶ using large-language models to improve group decision-making; and
- ▶ liquid democracy (time permitting).

PAC-Learning of Voting Methods

Ariel D. Procaccia, Aviv Zohar, Yoni Peleg, and Jeffrey S. Rosenschein (2009). *The learnability of voting rules*. Artificial Intelligence 173, pp. 1133 - 1149.

Yuval Salant (2007). *On the learnability of majority rule*. Journal of Economic Theory, 135, pp. 196 - 213.

Kanad Shrikar Pardeshi, Itai Shapira, Ariel D. Procaccia, and Aarti Singh (2024). *Learning Social Welfare Functions*. Working paper.

PAC-Learning of Voting Methods

Ariel D. Procaccia, Aviv Zohar, Yoni Peleg, and Jeffrey S. Rosenschein (2009). *The learnability of voting rules*. Artificial Intelligence 173, pp. 1133 - 1149.

Yuval Salant (2007). *On the learnability of majority rule*. Journal of Economic Theory, 135, pp. 196 - 213.

Kanad Shrikar Pardeshi, Itai Shapira, Ariel D. Procaccia, and Aarti Singh (2024). *Learning Social Welfare Functions*. Working paper.

Learning voting rules

- An entity, which is referred to as the designer, has in mind a voting rule (which may reflect the ethics of a society). It is assumed that the designer is able, for each constellation of voters' preferences with which it is presented, to designate a winning alternative (perhaps with considerable computational effort).

Learning voting rules

- An entity, which is referred to as the designer, has in mind a voting rule (which may reflect the ethics of a society). It is assumed that the designer is able, for each constellation of voters' preferences with which it is presented, to designate a winning alternative (perhaps with considerable computational effort).

- In particular, one can think of the designer's representation of the voting rule as a black box that matches preference profiles to winning alternatives. This setting is relevant, for example, when a designer has in mind different properties it wants its rule to satisfy; in this case, given a preference profile, the designer can specify a winning alternative that is compatible with these properties.

Learning voting rules

- The goal is to find a concise and easily understandable representation of the voting rule that the designer has in mind.

- *Automated design of voting rules*: given a specification of properties, or, indeed, of societal ethics, find an elegant voting rule that implements the specification.

Learning voting rules

- The goal is to find a concise and easily understandable representation of the voting rule that the designer has in mind.

- *Automated design of voting rules*: given a specification of properties, or, indeed, of societal ethics, find an elegant voting rule that implements the specification.

- Assume further that the "target" voting rule the designer has in mind, i.e., the one given as a black box, is known to belong to some family $\mathcal{R}$ of voting rules. We would like to produce a voting rule from $\mathcal{R}$ that is as "close" as possible to the target rule.

Introduction to PAC-Learning

- PAC (Probably Approximately Correct) learning is a theoretical framework for understanding the feasibility of learning.
- It aims to define the conditions under which a learner can learn a function that generalizes well from a limited set of training examples.
- The learner tries to find a hypothesis h from a hypothesis class H that approximates a target function f^*.
- The goal is to ensure that with high probability, the hypothesis has an error less than a specified threshold ϵ.

Realizable Case

- The learner is attempting to learn a function $f : Z \to Y$, which belongs to a class $\mathcal{F}$ of functions from Z to Y.

Realizable Case

- The learner is attempting to learn a function $f : Z \to Y$, which belongs to a class $\mathcal{F}$ of functions from Z to Y.
- The learner is given a training set—a set $\{z_1, \ldots, z_t\}$ of points in Z, which are sampled i.i.d. (independently and identically distributed) according to a distribution D over the sample space Z. D is unknown, but is fixed throughout the learning process.

Realizable Case

- The learner is attempting to learn a function $f : Z \rightarrow Y$, which belongs to a class $\mathcal{F}$ of functions from Z to Y.
- The learner is given a training set—a set $\{z_1, \ldots, z_t\}$ of points in Z, which are sampled i.i.d. (independently and identically distributed) according to a distribution D over the sample space Z. D is unknown, but is fixed throughout the learning process.
- Assumption: A target function f^* exists such that training examples are

$$\{(z_k, f^*(z_k))\}_{k=1}^{t}.$$

Realizable Case

- The learner is attempting to learn a function $f : Z \to Y$, which belongs to a class $\mathcal{F}$ of functions from Z to Y.
- The learner is given a training set—a set $\{z_1, \ldots, z_t\}$ of points in Z, which are sampled i.i.d. (independently and identically distributed) according to a distribution D over the sample space Z. D is unknown, but is fixed throughout the learning process.
- Assumption: A target function f^* exists such that training examples are

$$\{(z_k, f^*(z_k))\}_{k=1}^{t}.$$

- Error of a function $f \in \mathcal{F}$:

$$\mathrm{err}(f) = \Pr_{z \sim D}[f(z) \neq f^*(z)]$$

Accuracy and Confidence

- Accuracy parameter $\epsilon > 0$: Desired accuracy of the learning process.
- Confidence parameter $\delta > 0$: Probability that error exceeds ϵ:

$$\Pr[\mathrm{err}(h) > \epsilon] < \delta$$

PAC-Learning Formalization

- A learning algorithm L is a function that maps training examples to functions in $\mathcal{F}$ with the following property:

PAC-Learning Formalization

- A learning algorithm L is a function that maps training examples to functions in $\mathcal{F}$ with the following property: given $\epsilon, \delta \in (0, 1)$, there exists an integer $s(\epsilon, \delta)$ (the *sample complexity*)

PAC-Learning Formalization

- A learning algorithm L is a function that maps training examples to functions in $\mathcal{F}$ 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 $s(\epsilon, \delta)$ where the samples are drawn i.i.d. according to D, then with probability at least $1 - \delta$ it holds that $err(L(Z)) \leq \epsilon$.

PAC-Learning Formalization

- A learning algorithm L is a function that maps training examples to functions in $\mathcal{F}$ 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 $s(\epsilon, \delta)$ where the samples are drawn i.i.d. according to D, then with probability at least $1 - \delta$ it holds that $err(L(Z)) \leq \epsilon$.

- 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 Z, and of elements in Y.

PAC-Learning Formalization

- A learning algorithm L is a function that maps training examples to functions in $\mathcal{F}$ 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 $s(\epsilon, \delta)$ where the samples are drawn i.i.d. according to D, then with probability at least $1 - \delta$ it holds that $err(L(Z)) \leq \epsilon$.

- 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 Z, and of elements in Y.

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

Theorem

The class of scoring rules for n voters and m candidates efficiently PAC-learnable.

Ariel D. Procaccia, Aviv Zohar, Yoni Peleg, and Jeffrey S. Rosenschein (2009). *The learnability of voting rules*. Artificial Intelligence 173, pp. 1133 - 1149.

Theorem

The class of scoring rules for n voters and m candidates efficiently PAC-learnable.

Ariel D. Procaccia, Aviv Zohar, Yoni Peleg, and Jeffrey S. Rosenschein (2009). *The learnability of voting rules*. Artificial Intelligence 173, pp. 1133 - 1149.

"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 alternatives. 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."

for $k \leftarrow 1 \ldots s$ **do**
  $X_k \leftarrow \emptyset$
  for all $x_j \neq x_{j_k}$ **do** ▷ x_{j_k} is the winner in example k
    $\vec{\pi}^{\Delta} \leftarrow \vec{\pi}^{k}_{j_k} - \vec{\pi}^{k}_{j}$
    $l_0 \leftarrow \min\{l\colon\ \pi^{\Delta}_{l} \neq 0\}$
    if $\pi^{\Delta}_{l_0} < 0$ **then** ▷ Ties are broken in favor of x_j
      $X_k \leftarrow X_k \cup \{x_j\}$
    end if
  end for
end for
return a feasible solution $\vec{\alpha}$ to the following linear program:

$$\boxed{\begin{array}{l}
\forall k,\ \ \forall x_j \in X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l + 1 \\
\forall k,\ \ \forall x_j \notin X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l \\
\forall l = 1, \ldots, m-1 \quad \alpha_l \geqslant \alpha_{l+1} \\
\forall l,\ \ \alpha_l \geqslant 0
\end{array}}$$

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

for $k \leftarrow 1 \ldots s$ **do**

$X_k \leftarrow \emptyset$

for all $x_j \neq$ ... ner in example k

$\vec{\pi}^{\Delta} \leftarrow \vec{\pi}$ ...

$l_0 \leftarrow$ mi ...

if $\pi^{\Delta}_{l_0} <$ ... en in favor of x_j

$X_k \leftarrow$ ...

end if

end for

end for

Deal with the problem of ties:

x_{j_k} is the winner in election k.

$a \in X_k$ means that ties between a and x_{j_k} are broken in favor of a

return a feasible solution $\vec{\alpha}$ to the following linear program:

$$\boxed{\begin{array}{l}
\forall k,\ \forall x_j \in X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l + 1 \\
\forall k,\ \forall x_j \notin X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l \\
\forall l = 1, \ldots, m-1 \quad \alpha_l \geqslant \alpha_{l+1} \\
\forall l,\ \alpha_l \geqslant 0
\end{array}}$$

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

```
for k ← 1...s do
    X_k ← ∅
    for all x_j ≠ x_{j_k} do                          ▷ x_{j_k} is the winner in example k
        π^Δ ← π^k_{j_k} − π^k_j
        l_0 ← min{l: π^Δ_l ≠ 0}
        if π^Δ_{l_0} < 0 then                        ▷ Ties are broken in favor of x_j
            X_k ← X_k ∪ {x_j}
        end if
    end for
end for
return a feasible solution α to the following linear program:
```

$$
\boxed{\begin{array}{l}
\forall k,\ \forall x_j \in X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l + 1 \\
\forall k,\ \forall x_j \notin X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l \\
\forall l = 1, \ldots, m-1 \quad \alpha_l \geqslant \alpha_{l+1} \\
\forall l,\ \alpha_l \geqslant 0
\end{array}}
$$

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

for $k \leftarrow 1 \ldots s$ **do**
  $X_k \leftarrow \emptyset$
  for all $x_j \neq x_{j_k}$ **do** ▷ x_{j_k} is the winner in example k
    $\vec{\pi}^{\Delta} \leftarrow \vec{\pi}^{k}_{j_k} - \vec{\pi}^{k}_{j}$
    $l_0 \leftarrow \min\{l\colon\ \pi^{\Delta}_{l} \neq 0\}$
    if $\pi^{\Delta}_{l_0} < 0$ **then** ▷ Ties are broken in favor of x_j
      $X_k \leftarrow X_k \cup \{x_j\}$
    end if
  end for
end for

return a feasible solution $\vec{\alpha}$ to the following linear program:

$\forall k,$

$\forall k,$

$\forall l = 1, \ldots, m-1 \quad \alpha_l \geqslant \alpha_{l+1}$

$\forall l,\ \alpha_l \geqslant 0$

$\pi^{k}_{j,l}$: The number of voters that rank x_j in position l in election k

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

for $k \leftarrow 1 \ldots s$ **do**
  $X_k \leftarrow \emptyset$
  for all $x_j \neq x_{j_k}$ **do** ▷ x_{j_k} is the winner in example k
    $\vec{\pi}^{\Delta} \leftarrow \vec{\pi}^{k}_{j_k} - \vec{\pi}^{k}_{j}$
    $l_0 \leftarrow \min\{l\colon\ \pi^{\Delta}_{l} \neq 0\}$
    if $\pi^{\Delta}_{l_0} < 0$ **then** ▷ Ties are broken in favor of x_j
      $X_k \leftarrow X_k \cup \{x_j\}$
    end if
  end for
end for
return a feasible solution $\vec{\alpha}$ to the following linear program:

$$\boxed{\begin{array}{l} \forall k,\ \forall x_j \in X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l + 1 \\ \forall k,\ \forall x_j \notin X_k, \sum_l \pi^k_{j_k,l}\alpha_l \geqslant \sum_l \pi^k_{j,l}\alpha_l \\ \forall l = 1, \ldots, m-1 \quad \alpha_l \geqslant \alpha_{l+1} \\ \forall l,\ \alpha_l \geqslant 0 \end{array}}$$

$\forall a \in X_k,\ Score(x_{jk}) > Score(a)$
$\forall a \notin X_k,\ Score(x_{jk}) \geq Score(a)$
It is a sensible scoring rule

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

Given examples that are consistent with some general voting rule, is it possible to learn a scoring rule (or a small voting tree) that is “close” to the target rule?

Given examples that are consistent with some general voting rule, is it possible to learn a scoring rule (or a small voting tree) that is "close" to the target rule?

Fix n voters and m candidates. A voting method F is a c-approximation of a voting rule G provided that F and G agree on a c-fraction of the profiles:

$$|\{\mathbf{P} \mid F(\mathbf{P}) = G(\mathbf{P})\}| \geq c \times (m!)^n.$$

Theorem (Procaccia et al. 2009)

Let $\mathcal{R}_m^n$ be a family of voting rules of size exponential in n and m, and let $\epsilon, \delta > 0$. For large enough values of n and m, at least a $(1-\delta)$-fraction of the voting rules F satisfy the following property: no voting rule in $\mathcal{R}_m^n$ is a $(1/2+\epsilon)$-approximation of F.

Corollary (Procaccia et al. 2009)

For large enough values of n and m, almost all voting rules cannot be approximated by a scoring rule on n and m to a factor better than $1/2$.