Machine Learning? In *My* Election? It's More Likely Than You Think:

<u>**Voting Rules via Neural Networks**</u>

Daniel Firebanks-Quevedo
Advisor: Sam Taggart

Voting

- UK General Elections 2019 - ⅕ people planned to vote strategically
- 2 main contenders, but...

<u>Problem:</u> You might be discouraged from voting for the candidate you like the most.

Unfortunately...

Gibbard-Satterthwaite Theorem

*For **3 or more candidates**, any rule that presumes a single winner and is **not dictatorial** is **not strategy-proof**.*

- Many impossibility theorems stating trade-offs in social choice
- Think of different notions of welfare...
  - ***Arrow's Impossibility Theorem***

- **General problem:** Welfare and non-manipulability seem to be mutually exclusive

Gibbard-Satterthwaite Theorem

*For **3 or more candidates**, any rule that presumes a single winner and is **not dictatorial** is **manipulable**.*

Universal case:

For any rule, there exists some setting for which not every desired property holds

(worst-case)

Specific case:

For a given setting, there could exist a rule that satisfies our desired properties.

(average-case)

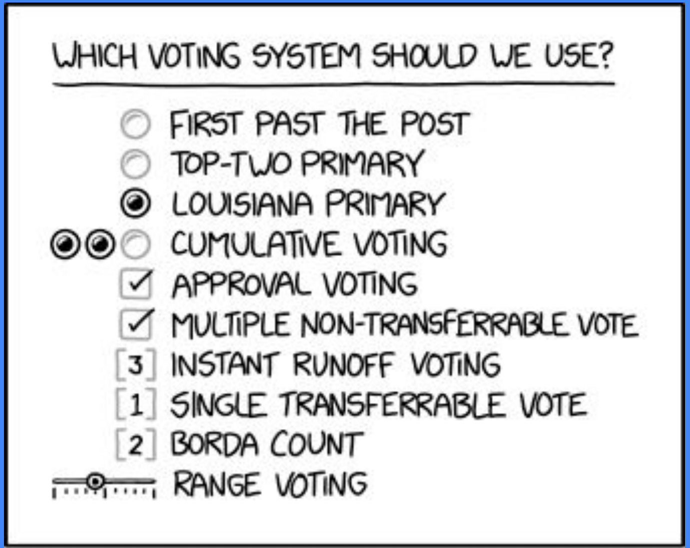
How do we deal with the tradeoff between welfare and non-manipulability?

How can we circumvent impossibility theorems for a particular setting?

Goal: ***Design a voting mechanism for the average case.***

Outline

- Previously in Social Choice
- Formal Setting
- Social Choice as a Learning Problem
- Proposed Framework
- AVNet
- Experiments



THE REFERENDUM WENT WELL, BUT WE CAN'T FIGURE OUT HOW TO COUNT THE BALLOTS.

Previously, in Social Choice…

<u>Preference Structure</u>

★ There are strategy-proof rules for single-peaked preferences

❏ Makes assumptions about preferences.

<u>Probabilistic Social Choice</u>

★ With randomness in the rule, we can satisfy some notions of non-manipulability!

❏ Non-democratic

Average case rule design

- **<u>Automated Mechanism Design</u>**: Constructing a rule is an optimization problem

Procaccia et al, 2009: Automated voting design

★ Rule is a black box that learns a mapping

❏ Rule has to be of a certain family

<u>Ideas:</u>

★ Rule is a black box that learns a mapping

★ Solve an optimization problem using machine learning

★ Use universal function approximators

Voting Mechanisms

- Set ***N*** of ***n*** voters
- Set **A** of ***m*** alternatives
- A ***ballot*** for each voter *i*
- A ***preference profile P***

$\boldsymbol{P} =$

4	2	1
b	*a*	*a*
a	*b*	*c*
c	*c*	*b*

A voting rule is a ***social choice function*** that maps ***sets of preference rankings*** to a ***particular winner(s)***.

$$f : \mathcal{P} \rightarrow S(A)$$

Borda

- Scores
- Winner: a

Copeland

- Pairwise comparisons
- Winner: h

4	3	3	3
a	*a*	*o*	*h*
o	*h*	*h*	*a*
h	*o*	*a*	*o*

What is a “good” voting rule?

- Welfare constraints

Condorcet complicity: *Choose the Condorcet winner if there is one.*

Majority criteria: *Choose the majority candidate if there is one.*

- Non-manipulability constraints

Individual Manipulation (IM): *A single voter can alter the outcome of the election by voting strategically*

How do we evaluate it?

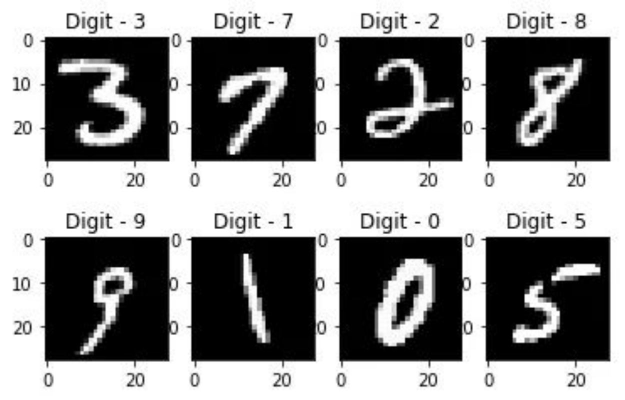
→ Samples from a distribution
→ Satisfy our constraints as much as possible

Goal

Given a particular distribution D and a set of constraints C, we want to find a voting rule that satisfies the constraints in C with high probability over preference profile P ~ D.

The Learning Behind Social Choice

- ***Classification task:*** A model learns a function h : X → Y that maps *inputs x* ∈ *X* to *labels y* ∈ *Y* .



- ***In social choice:*** For every *preference profile (our input)* there exists *a winner (a label)* that represents the best candidate according to the constraints we have previously defined.

Key difference: *We have constraints that the labeling system has to satisfy as opposed to "correct" labels.*

Framework

Choose a **distribution**

Choose your *favorite* **constraints**

Condorcet, majority, IM

Evaluation

Constraint satisfaction rate: #satisfied/#evaluated

Generate **preference profiles**

Neural Network

Stochastic Gradient Descent

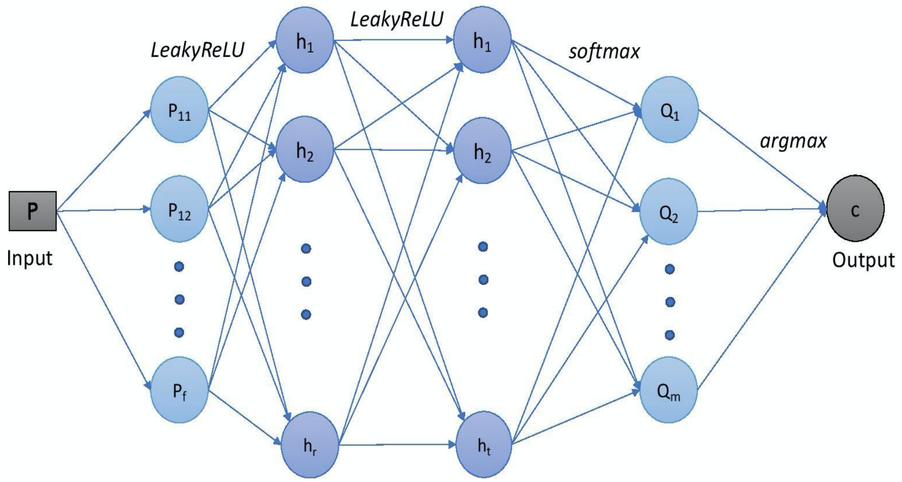
Testing

AVNet Design

- Input transformation

4	2	1
a	*b*	*a*
b	*c*	*c*
c	*a*	*b*

a	b	c
6	*3*	*0*
0	*4*	*5*
3	*2*	*4*



- **2 hidden layers** + **Dropout + softmax**
- *Multiple architectures*

Loss Function

Welfare loss

$$L_w(P) = \lambda_w \left[-\sum_{c_i \in A} p^*(c_i) \cdot \log(p(c_i)) \right]$$

Counterfactual loss

$$L_s(\mathcal{P}) = \lambda_s \frac{1}{|\mathcal{P}'|} \sum_{P' \in \mathcal{P}'} \left[-\sum_{c_i \in A} p(c_i) \cdot \log(p'(c_i)) \right]$$

Total loss

$$\mathcal{L}(\mathcal{P}) = \sum_{C_i \in C} L_{C_i}(P) = L_w(P) + L_s(P)$$

$$\mathcal{L}(\mathcal{P}) = \lambda_w \left[-\sum_{c_i \in A} p^*(c_i) \cdot \log(p(c_i)) \right] + \lambda_s \frac{1}{|\mathcal{P}'|} \sum_{P' \in \mathcal{P}'} \left[-\sum_{c_i \in A} p(c_i) \cdot \log(p'(c_i)) \right]$$

Experiments

	Setup 1	**Setup 2**
# of candidates	3	5
# of voters	20	40, 80
Distributions	Spheroid, Cubic, Ladder	
% of Condorcet, majority	~60%, ~30%	~40%, 0%

High Welfare, High IM rate

- Good balance between welfare and non-manipulability
- Difference in Condorcet rate was 1 candidate
- In the other distributions we didn't necessarily achieve the first best IM rate, but we did do second or third best
    - The baselines that performed best were different in each distribution

Voting Rule	**Condorcet rate**	**Majority rate**	**Plurality rate**	**Mean IM rate**	**Mean IM score**
RuleBorda	1.0	1.0	0.5	0.91	1.41
RuleMaximin	1.0	1.0	0.45	0.872	2.082
RuleCopeland	**1.0**	1.0	0.45	0.91	1.41
RuleCondorcet	1.0	1.0	0.5	0.808	3.089
RulePlurality	0.875	1.0	0.45	0.91	3.022
RuleSchulze	1.0	1.0	0.45	0.885	1.881
RuleBucklinInstant	1.0	1.0	0.45	0.885	1.881
RuleVeto	1.0	1.0	0.45	0.885	1.813
*AVNet**	0.9375	**1.0**	**0.556**	**0.936**	**0.592**

(a) Cubic, 3 candidates, 20 voters, architecture #5

Medium welfare, high IM rate

- Better than random but still not optimal welfare
- Either lack of training instances or wrong lambda parameters
  - *Latest update:* requires at least 60% of occurrences to learn
- Best performing baselines here are different across distributions as well

Voting Rule	**Condorcet rate**	**Plurality rate**	**Mean IM rate**	**Mean IM score**
RuleBorda	0.7	0.35	0.903	3.968
RuleMaximin	**1.0**	0.35	0.898	1.642
RuleCopeland	1.0	0.35	0.852	2.406
RuleCondorcet	1.0	0.2	0.578	6.801
RulePlurality	0.3	**0.35**	0.911	7.076
RuleSchulze	1.0	0.35	0.89	1.813
RuleBucklinInstant	0.7	0.3	0.886	4.231
RuleVeto	0.3	0.25	0.94	6.594
*AVNet**	*0.4*	*0.2*	**0.962**	**0.216**

(c) Spheroid, 5 candidates, 80 voters, architecture #1

Conclusions and Future Work

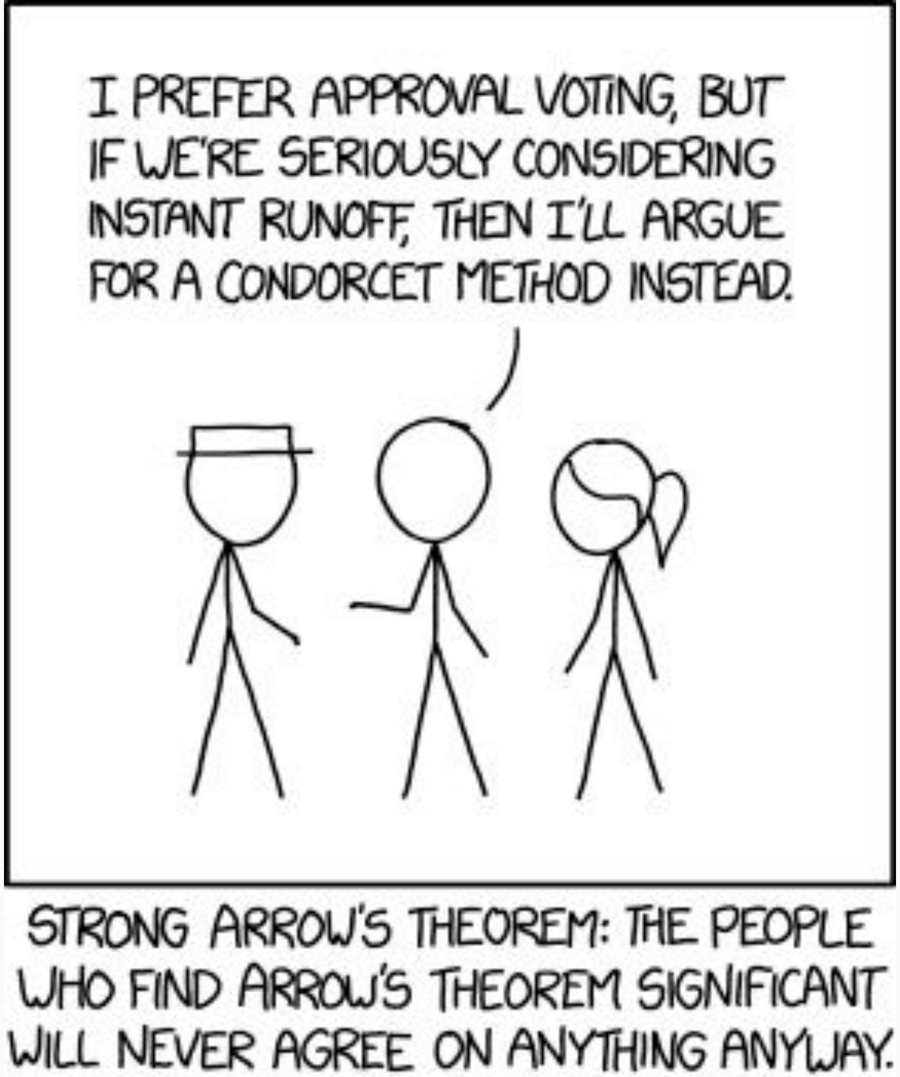
1. With enough data, our neural network can learn to effectively trade-off between welfare and non-manipulability constraints.
2. Unfortunately, in scenarios with high candidate-to-voter ratio, welfare performance declines if not enough examples are given.
3. *Natural objection:* Why would we use a black-box?

- New ways of manipulation
- Scaling up number of voters
- Compare against other baselines
- Hyperparameter tuning
- More data + optimizing data generation

Acknowledgements

- Sam Taggart, Adam Eck
- Jane Hsieh, Charles Cui, Tumas Račkaitis, Sam Barr

Thank you


I PREFER APPROVAL VOTING, BUT IF WE'RE SERIOUSLY CONSIDERING INSTANT RUNOFF, THEN I'LL ARGUE FOR A CONDORCET METHOD INSTEAD.
STRONG ARROW'S THEOREM: THE PEOPLE WHO FIND ARROW'S THEOREM SIGNIFICANT WILL NEVER AGREE ON ANYTHING ANYWAY.