Machine Learning to Strengthen Democracy

Ben Armstrong, Kate Larson

David R. Cheriton School of Computer Science, University of Waterloo

Motivation

Voting systems tend to have flaws such as vulnerability to strategic voting or bias towards particular types of winners (e.g. plurality voting often prefers more extreme candidates). Arrow's Theorem shows that all voting systems will occasionally have undesirable outcomes.

Good voting rules should provide reasonable results when most voters are behaving in a manner that could be considered reasonable.

There are many competing criteria/axioms used to evaluate voting rules. Different axioms may be suitable in different domains. Computing whether axioms are met can be very slow.

We develop a machine learning tool to find the best candidates based on a set of axioms chosen to suit a particular setting. While it cannot behave perfectly, this tool should meet the chosen axioms in the majority of elections.

Axioms

Choose a set of axioms that reflect the type of result desired for the current domain.

Generate Election Data

Generate sample ballots $b \in B$ that, when combined with a winning candidate $c \in C$, exemplify a particular axiom.

Define Scoring Function

Define a function $S : B \times C \to \mathbb{R}$ giving a score for each (ballot, winner) pair. Give higher scores to more ideal winners.

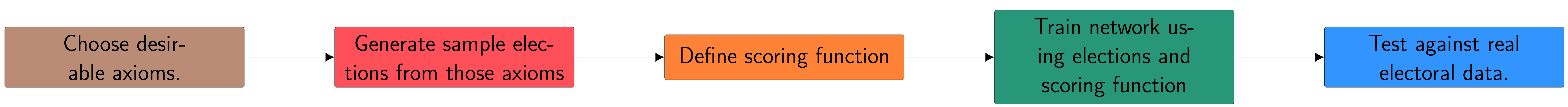
Train Network

Use $B \times C$ and S to train the network to predict a score for each candidate in unseen elections.

Test Against Real Data

Use data from recent Canadian elections. For each riding, compare performance of trained network with the outcome from actual voters.

Term	Definition
Condorcet Criterion	An axiom that is met when, if a candidate beats all other candidates in pairwise elections they are the winner.
Plurality Rule	A common voting rule in which the candidate with the most first choice votes wins.



Training

We train networks to simulate the 6 major political groups in Canada. A sample election with three voters and no Condorcet winner is shown below. The preference matrix is one component of a training sample, along with a candidate and the score if that candidate were to win.

$$v_1 : c_1 \succ c_2 \succ c_3 \succ c_4$$
$$v_2 : c_2 \succ c_3 \succ c_1 \succ c_4$$
$$v_3 : c_3 \succ c_1 \succ c_2 \succ c_4$$

Preferences of three sample voters that, when paired with a scoring function, help to teach the system how to behave when there is no Condorcet winner.

Candidate	c_1	c_2	c_3	c_4
c_1	0	0.66	0.33	1.0
c_2	0.33	0	0.66	1.0
c_3	0.66	0.33	0	1.0
c_4	0	0	0	0

Pairwise preferences for each voter pair. For example, candidate 3 is preferred over candidate 1 by 2 of the 3 voters.

Initially, we teach our network to elect the Condorcet winner (if it exists). For this, we use the following score function and

$$S(b, c) = \begin{cases} 0 & \text{c is Condorcet winner,} \\ 0.5 & \text{c wins the most pairwise elections but} \\ & \text{no Condorcet winner exists,} \\ 1 & \text{otherwise} \end{cases}$$

Results

Training Data	Training Accuracy
X_{train}	0.76
$X_{c=p}$	0.99
$X_{c\neq p}$	0.91
$X_{\text{no c}}$	1.0
$X_{c=p} \cup X_{c\neq p}$	0.91

Training accuracy on various subsets of training data. We train separately with elections where the Condorcet winner is and is not the plurality winner, and where there is no Condorcet winner.

	Accuracy per Year			
Training Data	2006	2008	2011	2015
X_{train}	0.59	0.70	0.69	0.54
$X_{c=p}$	0.92	0.94	0.94	0.91
$X_{c\neq p}$	0.90	0.87	0.87	0.86
$X_{\text{no c}}$	0.38	0.47	0.52	0.35
$X_{c=p} \cup X_{c\neq p}$	0.80	0.84	0.78	0.75
Real Voter Accuracy	0.95	0.97	0.96	0.97

Testing accuracy using a model trained on different subsets of artificially generated data to predict the highest-scoring candidate according to S in each district of each federal election (not necessarily the actual winning candidate).

Discussion

- Our system identifies the ideal winner much more frequently than a random guess.
- Our training method **can adapt to teach many additional axioms**.
- Predicting a score for each candidate, rather than a single winner, allows trivial extension to multi-winner elections.
- With the current structure of our input data, some axioms are impossible to represent.
- Size of training data may increase significantly with each new axiom; election runners should choose small sets of axioms.
- Choosing different axioms based on the election domain leads to more unique and customizable voting rules.
- There is a tradeoff between being easy to analyze the rule and being able to understand the best manipulations for a rule.
- Unique voting rules are less easily analyzed and likely to be **more difficult to manipulate**.
- Lack of explainability/predictability may limit the areas where this technique can be responsibly used.