Machine Learning to Strengthen Democracy

Ben Armstrong
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON
ben.armstrong@uwaterloo.ca

Kate Larson
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON
kate.larson@uwaterloo.ca

Abstract

Democratic institutions making use of plurality voting have, in recent years, been shown to be susceptible to interference and to poorly reflect the interests of voters. While many agree that better systems exist, there is no clear consensus on how the perfect election system functions. This paper describes a machine learning approach to create a voting rule aimed at satisfying highly customisable sets of ideal electoral fairness criteria. We first provide a description of our framework, then show in a simple example that it is able to achieve a strong result. Our system aims to provide room for opposing groups to agree on the principles they believe an election should reflect, rather than arguing over pre-existing preferences for particular reforms.

1 Introduction

Most Western democracies rely on some form of plurality voting which is widely recognized as having many flaws despite being in such common usage. Among other issues, it encourages dishonest strategic voting, doesn't allow voters to express their preferences, and tends to reach a non-representative two-party system. These problems can even compound when elections occur over many districts.

These problems can be alleviated by switching to a more powerful voting system, however a major hurdle to accomplishing this is consensus in which system to adopt. Canadian Prime Minister Justin Trudeau chose not to pursue electoral reform stating that "consensus has not emerged" for which system to move towards [1]. A voting rule that acts based on non-partisan principles of fairness could go a long way towards addressing these issues.

In this paper we develop a system that selects a winning candidate based on a highly customisable set of desirable election criteria. Using standard machine learning techniques our system finds a balance between many, potentially mutually-exclusive, criteria that are chosen based on the goals of the election. The system operates with no external training data, only the selected criteria and promises to bring more representation to voter preferences. We begin by introducing the domain and background of this project, then describe the functionality of the system, and provide a simple example of a trained system at work.

2 Domain

While plurality voting is severely flawed, it turns out that no voting system can meet a set of reasonable fairness criteria [2]. Many axioms exist to evaluate potential voting systems such as Condorcet consistency (a winner must win pairwise elections against each other candidate), or Participation (voters are better off voting than abstaining). No system satisfies every axiom so finding

AI for Social Good workshop at NeurIPS (2019), Vancouver, Canada.

an optimal system becomes a problem of balancing which are more or less important for the particular context of the elections in question.

We focus in this paper on developing a replacement for a class of voting functions called "scoring" rules, in which each voter submits a ballot allotting a particular number of points to each candidate and the highest scoring candidate wins. Ballots in a plurality system give one point to a single candidate and no points to others, a ballot in a Borda count election with m candidates gives a single candidate m points, another $m-1$ points, and so on [3].

Similar work has been started by Xia which explores theoretical methods of integrating social choice axioms into a machine learning framework [4]. Xia considers several methods including the idea of incorporating axioms into training data and the error function, similar to our own approach. Previous research has frequently focused on using machine learning to compensate for missing or incomplete ballots rather than acting as the voting rule itself [5, 6], although Procaccia et al. start with a framework similar to ours and show that learning a scoring rule based on samples of output is theoretically possible [7]. Our work addresses a similar question with empirical results.

3 Methodology

Most common voting procedures such as plurality voting, Borda count, or approval voting have been created intuitively and are centuries old [3, 8]. In this paper we develop a framework for generating new, customisable voting rules with the goal of allowing an election designer to determine the set of (possibly mutually exclusive) axioms that they believe will lead to a fair outcome in their context. We use these axioms to generate training data for a neural network that, once trained, acts as the voting rule.

Election axioms: Axioms of voting rules define particular properties that may or may not hold based on the voting rule and outcome of an election. For instance, in order to meet the Condorcet criterion the winner of an election must be the candidate that wins pairwise elections against each other candidate if such a candidate exists. Plurality voting does not meet this axiom while less well-known rules such as the *Copeland* or *Kemeny-Young* methods [9] do satisfy the Condorcet criteria. Other well known axioms include *Independence of Irrelevant Alternatives* (the outcome of the election does not change if a non-winning candidate is added or removed) and *Participation* (voting honestly is always better for a voter than not voting at all) [9]. Some sets of axioms cannot all be satisfied simultaneously, our approach accepts any set of representable axioms and finds a winning candidate that strikes a balance between satisfying, or nearly satisfying, as many axioms as possible.

Training data: We are able to represent the practical definition of our selected set of axioms by pairs (B, c) where B contains a set of ballots and c is the winner that meets a particular axiom. If there is more than one possible winner we can train with multiple pairs (B, c_1), (B, c_2), etc. Each ballot contains the ordinal preferences of a single voter over all m candidates, e.g. $b_i = [0, 2, 1]$ indicates that voter i prefers candidate 0 to candidate 2 to candidate 1. To allow for an arbitrary number of voters we convert B into an $m \times m$ preference matrix E in which $E_{i,j}$ contains the fraction of voters that prefer candidate i to candidate j ($E_{i,i} = 0$). The entire training set consists of several pairs, one corresponding to each desired axiom.

While this structure ignores some information it is capable of representing a wide variety of voting rules. [1] The data used to train our model consists of pairs of: a set of ballots corresponding to each combination of axioms we wish to meet, and the ideal winning candidate for each ballot set. Each pair is repeated $m!$ times with candidates listed once in all possible orders to avoid biasing training to any particular candidate order.

Training procedure: We first define a penalty function $f(B, c)$ that returns a value for each candidate given a set of ballots and the candidate. f might correspond to the number of axioms that are not met or might be used to weight some axioms as more important than others and $c* = \arg\min_{c \in C} f(B, c)$ then corresponds to the ideal winner of the election. We phrase this as a regression problem and train a regression neural network to predict the score/penalty of each candidate then choose the lowest scoring candidate as the winner. This also suggests a straightforward extension to multi-winner elections: select the k lowest scoring candidates (though we don't explore this possibility in our evaluation).

[1] This includes, but is not limited to, rules within Fishburn's C1 and C2 classifications [10].

Testing Data: We use data from the Canadian federal elections in 2006, 2008, 2011, and 2015 made available by [11] to evaluate our system. In Canada, there are 308 (338 in 2015) districts (or, "ridings"). In each, voters that reside in that district elect a single representative using plurality voting. Districts have, on average, 228 polling stations. Each year's dataset records approximately 15 million votes, listing the polling station they were cast at and the candidate/party for which they were cast. The party that wins the plurality vote in the most districts wins the election. For simplicity and to limit computational requirements we limit our focus to 6 groups, consisting of the 5 parties that have won at least one seat in a recent election and a group merging all votes for other parties. Our categories are then Green, NDP, Liberal, Conservative, Bloc Québécois, and Other.

4 Experiments

We can evaluate the design of our framework by implementing it with some set of axioms and testing how accurately the trained network can select a candidate with a minimum penalty. We can also measure how well our system is able to simulate a real election by feeding it data from real Canadian elections. A dramatic departure from existing results should be considered undesirable without clear reason.

We begin by selecting a set of axioms we consider desirable. This can be a highly subjective process so we restrict our focus to a system that simply attempts to select the Condorcet winner when possible, or the candidate that maximizes the number of pairwise victories otherwise. We define our penalty function f to optimize for a Condorcet winner or to find a "near-Condorcet" winner when one doesn't exist:

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

Using $m = 6$ candidates (to match the Canadian election data) we construct training data X_{train} consisting of three components: $X_{c=p}$ is a set of ballots and winning candidates in which there is a Condorcet winner and it is the same as the plurality winner. $X_{c\neq p}$ is a set of ballots and winning candidates in which there is a Condorcet winner and it is *not* the same as the plurality winner. $X_{\text{no c}}$ is a set of ballots and winning candidates in which there is no Condorcet winner. Using all 720 permutations of each election our total training data consists of only 2160 samples. This data is used to train a regression neural network with a single hidden layer, implemented in Keras [12].

When testing with Canadian election data we must convert the plurality election results into ballots of the appropriate format. The data consists of the number of votes for each candidate at each polling station in each district. We convert the result at each polling station into an ordinal ranking over the parties and let it represent the ballot of a single "voter."

5 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

Table 1: Training accuracy when training with each subset of training data.

Table 1 shows we are clearly able to learn to satisfy our chosen axiom in the cases where the Condorcet winner exists, though combining different data may require further development. We see that when training sets represent similar concepts ($X_{c=p}$ and $X_{c\neq p}$) they merge with reasonable accuracy. We can also use the results of each Canadian election to test each training set and see in Table 2 that

our system selects the penalty-minimizing candidate much more accurately than a random guess. Observe that when the training data consists of only samples in which the plurality winner is the Condorcet winner, our neural network results in accuracy very similar to the "accuracy" of the elected candidate in the actual election (shown in Table 3).

	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

Table 2: Testing accuracy using a model trained on artificial data to predict the winner in each district of each federal election.

True Accuracy per Year			
2006	**2008**	**2011**	**2015**
0.95	0.97	0.96	0.97

Table 3: The frequency with which the true elected candidate in a real district is the candidate that minimizes our penalty function.

6 Discussion

Our proof-of-concept results show that we are able to construct a system for determining the winner of an election that meets some desirable axioms. Additional training data or alternate network structures could lead to higher accuracy and result in a more realistically usable system. Using machine learning to generate a customised election rule allows for experimentation with entirely novel combinations of ideal outcomes that has not been possible before. For example, this could enable election runners in one setting to develop an axiom that tries to incentivise larger numbers of parties to ensure all viewpoints are represented while in another setting the ideal election is meant to approximate some existing ground truth so using axioms that motivate honest, rather than strategic, voting can be used as the basis of the system.

A system such as ours is a unique contribution to the recent research focused on algorithmic generation of voting rules, such as [13]. Having an unbounded number of unique voting systems makes manipulation of the system more difficult while continuing to allow for reasoning about the system's properties. Though a notable limitation with our current approach lies in the structure of the data used to train the network. The preference matrix E is able to represent many useful axioms but some data from the original ballots is lost, making it impossible to fully recreate, e.g. the dynamics of a Single Transferable Vote election. Moving forward finding a more representative data structure that can recreate all ballot information in a size that does not depend on the number of voters, or a network structure that can incorporate ballots without biasing any particular outcome, would strengthen the flexibility of the system.

6.1 Responsible use

While our current system shows promising evaluation results and it might be improved by modifying the training procedure or data structure there are currently a limited number of areas where it could be responsibly used. Since the system, as with most neural networks, is not fully explainable there is only a statistical guarantee that it will not produce what humans would consider an obviously "wrong" winner. Thus, its use should be restricted to settings where the winner can do limited harm, or where there are limitations upon who can be a candidate.

The benefit of using this system is that in a small scale there may be a clear pattern of voting or outcome that the electorate universally recognizes as being desirable and which can be easily incentivised.

References

[1] Joanna Smith. Trudeau abandons promise for electoral reform, Feb 2017. URL https://www.macleans.ca/politics/ottawa/trudeau-abandons-promise-for-electoral-reform/.

[2] Mark Allen Satterthwaite. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory*, 10(2):187–217, 1975.

[3] Peter Emerson. The original borda count and partial voting. *Social Choice and Welfare*, 40(2): 353–358, 2013.

[4] Lirong Xia. Designing social choice mechanisms using machine learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 471–474. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[5] John A Doucette, Kate Larson, and Robin Cohen. Conventional machine learning for social choice. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[6] Lirong Xia and Vincent Conitzer. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research*, 41:25–67, 2011.

[7] Ariel D Procaccia, Aviv Zohar, Yoni Peleg, and Jeffrey S Rosenschein. The learnability of voting rules. *Artificial Intelligence*, 173(12-13):1133–1149, 2009.

[8] Josep M Colomer and Iain McLean. Electing popes: approval balloting and qualified-majority rule. *Journal of Interdisciplinary History*, 29(1):1–22, 1998.

[9] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

[10] Peter C Fishburn. Condorcet social choice functions. *SIAM Journal on applied Mathematics*, 33(3):469–489, 1977.

[11] Elections Canada. Elections Canada's Official Reports. http://www.elections.ca/content.aspx?section=res&dir=rep/off&document=index&lang=e, 2019. Accessed: 2019-02-25.

[12] François Chollet et al. Keras. https://keras.io, 2015.

[13] Nic Wilson. Generating voting rules from random relations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2267–2269. International Foundation for Autonomous Agents and Multiagent Systems, 2019.