Crowdsourcing Moral AI in Self-Driving Cars

Duke University

Jeremy Fox

December 4, 2016

Abstract

The thesis concerns the problem of moral decision making in self-driving cars. I examine some of the key philosophical issues involved in programming cars to make moral decisions and propose a proxy voting scheme as a solution. I begin by surveying related work in the topics of social choice, algorithmic morality and self-driving cars. I continue by exploring the problems inherent in this approach. I model the problem using a hierarchical model and give a limited bound on the probability of success via uniform random sampling. Finally, I conclude by suggesting further research avenues in this area.

Acknowledgements

I want to thank Professors Vincent Conitzer, Walter Sinnott-Armstrong, Jana Schaich Borg, and the rest of the Moral AI group for their continued help in writing this thesis.

Contents

Chapter 1

Introduction

The current state of artificial intelligence mandates the need to imbue machines with the ability to make moral decisions. Autonomous vehicles (AVs) roam our streets[5, 10], algorithms allocate kidneys to the sick, and within the next few years, we may see the arrival of AI used to target drone strikes and even sentence criminals. Thus, it is imperative that we as a society take steps towards programming and regulating these machines in ways that preserve our moral values.

Chapter 2

Review of Related Research

2.1 Social Choice Theory

Here, I briefly review the basics of social choice theory as they relate to my thesis. For a more in-depth understanding of social choice theory and its applications in computer science and AI, I refer readers to the Handbook of Computational Social Choice [2].

2.1.1 Social choice theory basics

Voting informally seeks to aggregate the preferences of a group of people. More formally, if we have N voters and M candidates, we define a *ballot* as a ranked ordering over alternatives by a voter, and a *voting rule* as a function that maps from the set of ballots to a single ranked ordering.

Often, in social choice theory, we are handed a set of ballots – the question is how to choose the best voting rule to aggregate them.

2.1.2 Votes as noisy perceptions of correct rankings

Although many people view vote aggregation as a way of selecting the candidate that best agrees with the preferences of voters, this is not the only model of voting. In 1785, Condorcet proposed a model of social choice in which a correct ranking existed over all the candidates, and our votes were noisy perceptions of this correct ranking [3]. In Condorcet's model, voters were more likely than not to make a correct ranking rather than an incorrect ranking, they cast their votes independently of one another, and they were

all equally likely to be correct. In this model, the question becomes: Which voting algorithm chooses a ranking that has maximal probability of being correct? Condorcet's Jury Theorem tells us that majority rule is the best decision function to use in this case.[3]

Theorem 2.1.1 (Condorcet Jury Theorem)**.** *Assume we have N voters voting over 2 alternatives for which their exists a correct ranking. Each alternative has an a priori chance of .5 of being correct, and each voter has a probability* $.5 < p_i \leq 1$ *of being correct. Let* P_N *denote the probability that majority vote over the groups preferences gives the correct decision.*

$$\lim_{N \to \infty} P_N = 1$$

This model can be extended to cases in which voters do not have equal probability of being right [6, 8].

Theorem 2.1.2 (The Bayesian Optimal Decision Rule)**.** *If we have a dichotomous decision with each choice having an a priori probability of .5 of being correct, and voters cast their votes independently of each other, then weighted majority vote maximizes the probability of being correct, with weights* w_i *given by:*

$$w_i \propto \log \frac{p_i}{1 - p_i}$$

2.2 Artificial Intelligence Morality

In his book *I, Robot*, Isaac Asimov recognized the problem of programming morality into machines and explored the ways in which humanity's attempts to solve this problem could go awry[1]. Today, AI morality is no longer the worry of science fiction – the potential benefits and dangers associated with artificial intelligence have been recently been highlighted by major news outlets (cite) and even the White House[4, 7]. However, as the One Hundred Year Study on Artificial Intelligence recently acknowledged, the societal impacts and safety of AI are currently under-researched and under-funded[9].

Chapter 3

Description of Main Problem

3.1 The Autonomous Vehicle Trolley Problem

3.1.1 Problem description

Many people have loosely and casually talked about the AV trolley problem over the past year, usually saying something along the lines of "should your self-driving car kill you, or others?" Although the problem of programming an AV to make moral decisions is very nuanced and involves consideration of varied, wide-ranging scenarios, I here formalize a more narrowed definition of the problem for use in my thesis.

The problem I will work with is as follows: an AV is driving on the road, with passengers inside, and encounters a situation in which there are people on the road. In this situation, there will be unavoidable harm that must come to either the passengers inside the car, or the pedestrians in front of the car. Since this harm is unavoidable, the car cannot deal with the question of how to avoid harm, but instead must decide who to harm. The car has two options – it can either drive straight, and hit the pedestrians in front of it, or swerve off the road, injuring its passengers. The car is presented with profiles of information on its passengers and the pedestrians – it must use this information to decide whether to swerve, or drive straight.

Definition 3.1.1 (Individual Profile)**.** *The information available about an individual, or their individual profile, is a vector composed of real numbers, categorical data, and boolean values.*

Example Individual Profile 3.1: Small boy

Age	Gender	Ran in front of car
7	Male	True

Example Individual Profile 3.2: Elderly driver

Age	Gender	Organ donor
82	Female	False

Definition 3.1.2 (The AV Trolley Problem)**.** *An AV is given two choices – drive straight, or swerve. There is a set of people inside the car, and a set of people in front of the car. The AV is guaranteed that to drive straight is to injure the parties in front of it, and to swerve is to injure the people inside the car. Given a set of individual profiles on the parties in front of the car and those inside the car, the AV trolley problem is to decide whether to drive straight or swerve.*

Although this problem has been widely discussed, few proposals have been made toward actually solving it. My original proposal was to use voting to solve this problem – in essence, to crowdsource a solution. Here we are faced with a problem – it is ridiculous to imagine individuals can actually vote on what decision an AV can make, as the AV will only have a few seconds to make such a decision – obviously, this is not enough time to elicit votes. Instead, I propose the use of *proxy voting algorithms* (equivalently, just voting algorithms) that can, in the inability of an individual to cast a vote, vote in their place.

Definition 3.1.3 (Voting algorithm)**.** *Let us denote the set of individual profiles of individuals inside the car and outside the car as S. Let's denote the action of "drive straight" as 1, "no decision" as 0, and "swerve" as -1. We call f a voting algorithm if $f : S \to \{1, 0, -1\}$.*

If we have voting algorithms, then we can imagine a new scenario: each individual submits a voting algorithm to a self driving car. The car's new

objective becomes to use these voting algorithms to decide what decision to make.

Now, I would like to introduce one more constraint – the AV does not necessarily have enough time to evaluate all voting algorithms. Why is this? It seems reasonable to assume that such a system, deployed on a society-wide scale, might contain hundreds of millions of voting algorithms, while an AV might only have a few tenths of a second in which to make a decision. Thus, the challenge becomes to evaluate the voting AV problem under a fixed time constraint.

Of course, this is still a relatively large and ambiguous problem, with wide-ranging constraints and criteria. To narrow this problem down for my thesis, I made several reasonable assumptions to turn this into a manageable problem – related problems are discussed extensively in the future research section.

Following is a list of the major assumptions I have made in defining this problem:

- Each person will submit one voting algorithm. Voting algorithms may take different forms – thus, while you may use a decision tree, I may map each scenario to a point in some Cartesian space, after which it is classified by a linear separator.
- Since each voting algorithm has a different form, different voting algorithms may take different amounts of time to run. Although algorithm speed may vary, we can sample each algorithm repeatedly to obtain a distribution over the time it takes to run.
- There may exist correlations between the runtime of a voting algorithm, and the probability it makes the correct decision.

From here, we can derive a new version of the AV voting problem problem

Definition 3.1.4 (The AV Voting Budget Problem (AVVBP))**.** *Given a budget, $\mathcal{B}$, a set of voting algorithms, V, where each voting algorithm v_i has associated cost c_i, and a set of individual profiles, S, and assuming there exists a correct decision Y, we wish to choose a decision $D \in \{1, -1\}$ such that the probability that*

$$D = Y$$

is maximized, subject to

$$\sum_i c_i < \mathcal{B}$$

.

3.1.2 Statistical Modeling and Random Sampling

Hierarchical Model

What follows is my statistical model for AVVBP. Let's assume there is a correct choice $Y \in \{-1, 1\}$. Each voting algorithm v_i will vote for Y with probability θ_i, a probability which is drawn from a beta distribution, Beta (α, β), where we can think of this beta distribution as a prior on a voter's ability to vote correctly. Thus, the correctness of voter i is distributed Bernoulli(θ_i), where θ_i is distributed $Beta(\alpha, \beta)$. We will call the random variable that turns up 1 if voter i votes correctly and 0 otherwise X_i. Now, let's also assume each voting algorithm runs with a cost $c_i = b_i + noise$, where b_i represents some base cost for voting function i, and noise is distributed Normal(μ, σ^2). We want b_i to be related to θ_i, so we will say that $b_i = f(\theta_i)$, where we can choose f appropriately to model different interactions relations between b_i and θ_i. For instance, if we want cost to relate inversely to ability to vote correctly, we could pick $f(\theta_i) = A(1 - \theta_i)$, where A is some fixed constant.

Random Sampling

Now that I have derived my model, I will show some results associated with various random sampling approaches. I would like to be able to give the exact probability that, in a random sample of n voting functions, at least half of the functions vote for the correct answer. I have not been able to do that – however, I have been able to provide a small lower bound on the probability the majority will vote correctly, given a constant c, where every voting function votes correctly with probability at least c.

Let random variable $X = \sum_{i=1}^{n} X_i$. I would like to lower bound $P(X > \frac{n}{2})$. Although I have not been able to lower bound this quantity directly, I can lower bound it in terms of a third variable, c.

Theorem 3.1.1.

$$P\left(X > \frac{n}{2}\right) > \sum_{i=\lceil n/2 \rceil}^{n} \binom{n}{i} c^i (1-c)^{n-1} Q^n$$

where

$$Q = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_c^1 x^{\alpha-1}(1-x)^{\beta-1}dx, \quad c \in [0,1]$$

Proof. Let $c \in [0,1]$ denote a lower bound on each θ_j – thus, each voter will vote correctly with probability at least c. Clearly, $P(X > \frac{n}{2}) > P(X > \frac{n}{2}$ and $\theta_j > c \; \forall j) = P(X > \frac{n}{2} \mid \theta_j > c \; \forall j) P(\theta_j > c \; \forall j)$, where the inequality comes from taking away some cases where X may be greater than $\frac{n}{2}$. For any θ_j, $P(\theta_j > c)$ can be calculated directly by integrating the probability density function of Beta(α, β) – thus, $P(\theta_j > c) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_c^1 x^{\alpha-1}(1-x)^{\beta-1}dx$. I will denote this quantity as Q. The probability that all theta values are greater than c is thus given by Q^n. Now, $P(X > \frac{n}{2} \mid \theta_j > c \; \forall j) > P(X > \frac{n}{2} \mid \theta_j = c \; \forall j)$. Thus, let us assume each $\theta_j = c$ in order to lower bound this quantity. In this case, we end up with a binomial distribution on the value c, and sum all instances where the number of "positive" values is greater than $n/2$. Thus, $P(X > \frac{n}{2} \mid \theta_j > c \; \forall j) > P(X > \frac{n}{2} \mid \theta_j = c \; \forall j) = \sum_{i=\lceil n/2 \rceil}^{n} \binom{n}{i} c^i (1-c)^{n-1}$ – multiplying this by Q^n gives us the lower bound on the proof. □

Chapter 4

Future Research

4.1 Decision Function Aggregation

Although I chose to pursue a random sampling approach, an alternative approach could involve aggregating voting functions in such a way that n voting functions could be collectively queried significantly faster than they could be queried individually. These aggregation methods would involve exploiting the structure of the voting functions. Here, I briefly explore some separate aggregation strategies.

4.1.1 Decision Boundaries

If each different AV problem can be represented as a point in some Cartesian space, then a hyperplane in this space can be thought of as a voting function, where all the points on one side of the plane get a vote of one , and all the points on the other side get a vote of negative one. If the set of all voting functions contains only hyperplanes, we can exploit the geometric structure of this set to quickly query all decision functions. More specifically, we can compute the arrangement of these hyperplanes, where each cell in the arrangement contains the sum of all the ones (for all every plane below the cell that classifies these points as a one), and all the negative ones (for planes above the cell that classify it as a negative one). This combined sum gives us the relative voting difference for each point in every cell.

Assuming we have computed the arrangement of all the decision boundaries, use of the correct data structure should allow us to efficiently query any one point in this space. Unfortunately, computing the arrangement of

hyperplanes in $\mathbb{R}^d$ has been shown to be $O(n^d)$ – thus, any work on this approach will need to deal with this problem, presumably via a clever dimension reduction.

4.1.2 Decision Trees

Bibliography

[1] Isaac Asimov. *I, robot*, volume 1. Spectra, 2004.

[2] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jerome Lang, and Ariel D. Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[3] Marquis de Condorcet. Essay on the application of analysis to the probability of majority decisions. *Paris: Imprimerie Royale*, 1785.

[4] Scott Dadich. Barack obama, neural nets, self-driving cars, and the future of the world, 2016.

[5] Anthony Levandowski and Travis Kalanick. Pittsburgh, your self-driving uber is arriving now, 2016.

[6] Shmuel Nitzan and Jacob Paroush. Optimal decision rules in uncertain dichotomous choice situations. *International Economic Review*, 23(2):289–297, 1982.

[7] Preparing for the future of artificial intelligence, 2016.

[8] Lloyd Shapley and Bernard Grofman. Optimizing group judgmental accuracy in the presence of interdependencies. *Public Choice*, 43(3):329–343, 1984.

[9] Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Sarit Kraus, Kevin Leyton-Brown, David Parkes, William Press, AnnaLee Saxenian, Julie Shah, Milind Tambe, and Astro Teller. Artificial intelligence and life in 2030, 2016.

[10] The Tesla Team. All tesla cars being produced now have full self-driving hardware, 2016.