Time-robust Optimal and Fair Automated Decisions

Making automated decisions that are optimal over time with regard to both fairness and utility: An experimental study of fairness in automated decision makers and objective functions

Aleksander Wang-Hansen

Thesis submitted for the degree of
Master in Data Science
60 credits

Department of Informatics
Department of Mathematics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2020

Time-robust Optimal and Fair Automated Decisions

Making automated decisions that are optimal over time with regard to both fairness and utility: An experimental study of fairness in automated decision makers and objective functions

Aleksander Wang-Hansen

Time-robust Optimal and Fair Automated Decisions

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Fairness is an inherently important aspect of decision-making in general, and with automated algorithms increasingly making decisions impacting our lives it is of paramount importance that the decisions they make be fair according to some fairness criterion and that this criterion is open to scrutiny. This is important both to ensure people are treated fairly, but also to ensure that the public, in general sceptical of machines having power, doesn't stop the promising field of automated decision-making in it's tracks.

This thesis explores a Bayesian approach to optimizing decisions in settings where we have little data and high uncertainty. We seek to optimize the decisions with regard both to the utility of the decision-maker and to the decisions being fair. Finally we seek to take the decisions that give the optimal utility and fairness not only for the decisions taken at the present, but for all decisions taken now and in the future, by also placing a value on the information gained from our decisions.

The results we were seeking to replicate did not, and our hypothesised method for improving that method resulted in no improvement. We therefore offer a framework for exploring approaches related to methods for optimizing a combination of fairness and utility, and of different utility functions, along with some reflections on the failures of the methods tested and possible other approaches to test out going forward.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Christos Dimitrakakis. Firstly for your invaluable course introducing me both to automated decision-making and to the aspect of fairness therein. Secondly for all your guidance during the writing of this thesis, and for carefully crafting your input to allow me to find my own solutions to the problems I was facing.

I would also like to thank my co-supervisors, Magdalena Ivanovska and Fabio Massimo Zennaro for guiding me through my initial research phase, for pointing me in the direction of fairness from a causal perspective, and for always replying promptly and thoroughly when I asked for input and feedback.

A big thank you to the faculty at the University of Oslo Institutes of Mathematics and Informatics responsible for the Master of Data Science program, led by Geir Olve Storvik, for their great efforts in figuring out and creating such a holistic study programme covering the wide span of the nascent field of Data Science.

Furthermore I must express my deep appreciation for my employer, Kolonial.no, and my boss Nina Wahlberg, for always supporting me and never asking me to compromise between my thesis and my work, and for offering me the computing power I needed on their computing platform when my Google Cloud funding ran out. Thanks also to my supervisor Kjetil Åmdahl-Sævik for pointing out the qualities of the Ranger optimizer when I was stuck with unconverging policies in difficult objective functions.

Thanks to Google Cloud for providing me with computing power when my experimentation got serious and beyond the capabilities of home computers.

And last but not least, a big thank you to my family: To Sine for always encouraging me in my efforts, never complaining about the long nights and weekends I spent in front of the computer and for inspiring me to take on the effort in the first place. Thank you to aunt Anveig and all the grandparents for taking care of the children while I worked, and finally to Eivind and Askild, for making it all worthwhile.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

1.1 Why Automated Decisions

Every day a staggering number of decisions needs to be made. This ranges from personal decisions about minute details in a single persons life, to significant, strategic decisions made by private citizens, businesses, and organizations; or decisions about personalized recommendations for services. Already in 1954 there was evidence of cases where automated decision making was preferrable to human decision making (Meehl 1954) with simple statistical methods outperforming trained psychological clinicians in predicting patient outcomes. Since this controversial point was introduced the evidence in its support has grown (Grove et al. 2000).

These days there is more and more focus on decisions being data-driven with well-documented reasoning behind the decision being made. Part of the reason for this drive is the realization that people frequently balance information unevenly and, based on their own personal experiences, introduce unconscious biases in their decisions.

In the sphere of personalized recommendations, particularly by online businesses, the sheer number of decisions being made and the available time for a decision to be made excludes manual decision making.

Lately, automated decision making through machine learning has been seen by many as a solution to unfairness often prevalent in human decision making based on bias, subjectiveness and favouritism. In the past few years, however, it has been shown that there are many challenges in this regard also when it comes to automated decision making by computers. Reasons for this unfairness in naive algorithms for machine learning-based decision makers are many: There could be problems with the data used for training such as biased data, unbalanced data or cencored data (Chouldechova and Roth 2018), it could be due to uncertainty about the model itself (Dimitrakakis et al. 2017), or it could be due to biases in the objective function. It might even be due to unconscious or conscious biases in the way the world is modelled.

1.2 Fairness

In general, fairness in decision making is an inherently desirable trait as we see our world today. One reason for this is the ethical viewpoint that in our current society we think it is right and good when people are treated equally regardless of what group they belong to or what traits they possess. Another is that the legitimacy of any decision, and of any decision-making process, depends on how it is percieved, and right now the appearance of fairness in any decision is vital to its legitimacy. The perception of fairness is even more important for automated decision makers[1] than for humans as it appears that we tolerate more faults from human decision makers than from machines.

In order to measure the fairness of a decision, we need to define what constitutes a fair decision. So far, several definitions of fairness have been introduced. In order to structure these definitions we will perform four dichotomous splits:

- *Focus*: Is the focus on fairness relative only to outcome, or to outcome given some measure of quality or fit of the individuals in question. The latter is the "similar outcome for similar individuals" criterion from Dwork and Ilvento 2018.

  In using the 'similar outcome for similar individuals' it also becomes vital to define how we measure similarity. If we measure similarity, directly or indirectly, based on the very traits or group belongings that we wish to avoid discriminating on, then we are simply institutionalizing discrimination.

  If we focus instead only on the outcome, then any hiring process will not be able to use, say, university education, as a sorting criterion if completing a university education might be more prevalent in one group than in another.

- *Stochasticity*: Are we judging fairness based on the realized outcome, or on the expected outcome? In the first case a lottery is terribly unfair, in the second case it is perfectly fair. This is the *ex-post* and *ex-ante* fairness in Kearns, Roth, and Wu 2017 where they also include a third, *ex-interim*, variant between the two.

- *Individual vs group*: Are we looking at individual fairness or group fairness? Under group fairness the interest is rather that the link between actions and outcomes does not differ significantly between groups Under individual fairness it is difficult to use the conventional measures of fairness because you don't have two groups for whom you may examine some statistical properties. Kusner et al. 2017 introduced a novel way of approaching individual fairness in which they use counterfactual fairness using causal modelling., i.e., measuring how different the action would have been if an individual had belonged to a different group.

[1] By an automated decision maker we mean a machine capable of returning a decision to a problem given a set of inputs by applying some algorithm. This could be through the use of hard-coded heuristics, through statistical or machine learning, or so-called artificial intelligence. In principle, for the discussion here, it could equally well be a manually applied algorithm without the use of machines, except for the argument about speed and volume of decisions.

- *Symmetry (over time)*: Do we require decisions to be perceived as fair by the people exposed to them only as they happen, or also when the action has already taken place? If a sense of fairness means to not be treated worse than others, and we only require a decision to appear fair to the person currently being affected by the decision, then we can relax our fairness criteria to use asymmetric fairness where we can make decisions that are more beneficial to people over time. This is the "fairness-in-hindsight" of Gupta and Kamble 2018.

Individual fairness that depends solely on the outcome of a decision is often difficult to measure and impossible to satisfy unless we focus on the *expected outcome*. In many scenarios the only way to satisfy it completely would be to make the same decision for everyone. In other scenarios it may well be impossible. For example, we can not let everyone be prime minister. It could be solved by giving everyone an equal chance to be selected, i.e. a lottery, thus treating everyone the same, and every individual will have the same expected outcome.

One problem with focusing on group fairness is that we are limited to making one specific criterion to split individuals into groups, meaning we give no regard to all *other* possible criteria that might be used. If we split by ethnicity, do we then not care about fairness between genders? Sexual orientation? Religious affiliation? On the other hand if we want to satisfy group fairness based on every conceivable group split we quickly approach individual fairness.

If we only focus on the outcome in group fairness, it is fairly simple. We need to satisfy "demographic parity" Liu et al. 2018, i.e. each side of a division created by the decisions taken should have similar demographics. In order to measure fairness it is sufficient to see how large a deviation there is between the demographic makeup of the whole population and of each subset of the population the decision maker creates.

For group fairness, using the principe of "similar outcome for similar individuals", two popular definitions at the moment are *Calibration* and *Balance*.

Definition 1.2.1 (Kleinberg, Mullainathan, and Raghavan 2016)**.** Kleinberg, Mullainathan, and Raghavan (2016) defines these fairness criteria in the context of a classification problem where

> *the positive class* consists of the people who constitute positive instances, and *the negative class* consists of the people who constitute negative instances. For example, for criminal defendants, the positive class could consist of those defendants who will be arrested again within some fixed time window, and the negative class could consist of those who will not.
>
> Each person also belongs to one of two groups, labeled 1 or 2, and we would like our decisions to be unbiased with respect to the members of these two groups. In our examples, the two groups

could correspond to different races or genders, or other cases where we want to look for the possibility of bias between them.

Informally, risk assessments are ways of dividing people up into sets based on their feature vectors σ (potentially using randomization), and then assigning each set a probability estimate that the people in this set belong to the positive class. Thus, we define a risk assignment to consist of a set of "bins" (the sets), where each bin is labeled with a score v_b that we intend to use as the probability for everyone assigned to bin b.

- *Calibration within groups* requires that for each group t, and each bin b with associated score v_b, the expected number of people from group t in b who belong to the positive class should be a v_b fraction of the expected number of people from group t assigned to b.
- *Balance for the negative class* requires that the average score assigned to people of group 1 who belong to the negative class should be the same as the average score assigned to people of group 2 who belong to the negative class. In other words, the assignment of scores shouldn't be systematically more inaccurate for negative instances in one group than the other.
- *Balance for the positive class* symmetrically requires that the average score assigned to people of group 1 who belong to the positive class should be the same as the average score assigned to people of group 2 who belong to the positive class.

For our purposes we need a somewhat less context-specific definition, and we need one we can compute clearly. We therefore group the two balance criteria into one and translate these criteria into mathematical formulae using the concept of conditional independence.

Definition 1.2.2 (Dimitrakakis et al. 2017)**.** Let z denote the *group* from Definition 1.2.1, y denote the *result*, analogous to what in Definition 1.2.1 is denoted as the positive and negative classes, but extended to a regression setting, and a denote the action taken, similar to the *bin* used in Definition 1.2.1.

- *Calibration*: Under the notion of calibration, for a given action by the decision-maker, the outcome is independent of group affiliation. Thus if calibration is satisfied completely we would see

$$P(y, z|a) = P(y|a)P(z|a) \tag{1.1}$$

- *Balance*: Under the notion of balance, for a given outcome, the action that was taken should be independent of group affiliation. Thus if balance is satisfied completely we would see

$$P(a, z|y) = P(a|y)P(z|y) \tag{1.2}$$

These are both sensible notions of fairness, and one might argue that both ought to be satisfied, but Kleinberg, Mullainathan, and Raghavan 2016 shows that, except in two special and trivial circumstances, it is impossible to satisfy both.

A final consideration when attempting to quantify the fairness of decisions and the similarities of individuals is the danger of overweighing measurable quantities in favour of unmeasurable ones as pointed out by Green and Hu 2018.

Fairness over time

In addition to the above mentioned static definitions of fairness, we need to define an appropriate measure of dynamic fairness. When considering the temporal effect of fairness, Gupta and Kamble 2018 defines a fairness-in-hindsight, under which fairness is assymetric in time. Under this notion of fairness a decision only needs to be judged fair by the current subject of the decision, not previous subjects. Hence, under this notion of fairness, the decision-maker can treat people better over time without violating the fairness criterion, but not worse. In contrast to this notion, we might apply a static definition of fairness across everyone affected by the decisions of the decision maker, thus disregarding time. A third option is to consider fairness within isolated periods of time. Then, in order to judge the total fairness, there needs to be some accumulation of the deviation from fairness across all time periods. If this is simply cumulative, the effect will be similar to using a static definition across all time periods, but it could also discount across time to empasize present fairness.

How you include the temporal aspect into the fairness assessment has direct implications on which decisions are optimal. Gupta and Kamble 2018 deduce their Careful Exploration from their fairness-in-hindsight-criterion in which they begin treating people harshly while exploring and gradually relaxing their decisions as they learn to make better decisions in order to reduce regret. This approach has some drawbacks, namely, it will not work if the level of utility granted to the subject of the decision is relevant for the information we get back, and it assumes that a consensus can be created for this time-dependent asymmetric sense of fairness. It also assumes that it is possible to treat people harshly in general at one time-point and not in another timepoint. This is in theory possible in the case of lending, but in our scenario of school admissions it makes little sense. Both because the decision we make is dichotomous, so we can't admit someone just a little, leading to the aforementioned information scarcity resulting from harsh decisions, and also because it makes no sense for a school to admit almost nobody as a Careful Exploration strategy.

Another aspect of fairness when we include the temporal aspect is whether or not it is in the long-term interest of a sensitive group that decisions are changed in order to satisfy fairness criteria at the present time. In general, there are two changes that can be made to decisions in order to improve fairness in the case where the maximum utility decision is violating group fairness. Either one can increase the randomness in the policy to improve fairness, given that we are focusing on the expected rather than the realized outcome, or one

can use different criteria for different groups. As shown by Liu et al. 2018 there are circumstances where maximizing group fairness can be detrimental to the protected groups long-term well-being in addition to being detrimental to institution utility. The example they use is a lending scenario in which a protected group is granted loans with lower credit rating scores. If this threshold is set too low then people will be given loans that they cannot expect to repay, putting them potentially in a worse position than if they had not been given that loan.

Any attempt at creating an automated decision-maker is predicated on a model that can be used to predict the future. Good, unbiased predictions of the future are a requirement for good, unbiased decisions. Any model that we create will be an uncertain representation of the world, although the more data we have the lower the uncertainty in our model. Any decision-maker, and certainly any decision-maker attempting to make fair decisions must take this uncertainty into account. Using a decision-maker that takes the uncertainty estimates of the model into account improves learning the best decisions in settings with low data and high model uncertainty (Dimitrakakis et al. 2017).

Task-specific fairness considerations

In addition to the above mentioned fairness concerns which largely are applicable in general, there can also be concerns that are specific to the situation under consideration. In our case, of admissions data, there could for example be the case that a higher probability of admission for students with lower grades would carry a significant fairness violation when we are not basing the decision on the protected variable. In cases where the protected variable is taken into account and the percieved weaker group is given advantageous treatment there could be some acceptance of this and a small fairness violation. If, however, the admission grade is the only information available to the policy, and it grants a higher admission probability to a lower grade (perhaps because it has realized that this would admit more of the weaker group), this would likely not be seen as acceptable.

1.3 Contribution

This thesis seeks to reproduce the results of Dimitrakakis et al. 2017 showing that a model-based calculation of expected fairness and utility taking into account the full uncertainty distribution of the model will perform better than one simply based on the marginal of the model. It will also seek to extend these results by exploring the behaviour of this algorithm and another using back-induction (DeGroot 2004) in optimizing for maximum utility and fairness over time.

In our setting of tertiary school admissions the fairness-in-hindsight-concept is not particualrly useful due to the drawbacks listed above. Instead, we consider each group of applicants at a timepoint as a group for which we desire fair actions, and then we accumulate fairness over time.

The results from Dimitrakakis et al. 2017 have not been reproduced. On

the contrary, the marginal of the models has shown the best performance and it is in the setting where we only care about fairness where the different methods have the most similar results. We do, however, show a significant improvement of the model-based approach to calculating fairness and utility over the empirical calculations more commonly employed when training Neural Nets with Stochastic Gradient Descent optimizers in low-data settings.

We also provide a GitHub repository with the source code used during the experimentation in this thesis, which is modifiable to test out different strategies on different datasets.

1.4 Outline

This thesis includes an Introduction with some theoretical concepts and some considerations around fairness followed by a description of the main work of the thesis.

Chapter 2 gives an overview of the problem and the setting we are exploring before continuing with a more specific discussion on fairness and utility, which fairness definitions we have chosen to use and how we measure them in our setting. Finally, we conclude the chapter with a discussion on how to measure fairness and utility over time, and how we weigh fairness and utility against each other.

Chapter 3 deals with the specific setting in which we try to solve our problem. In this chapter we explain the logic behind and the structure of the data generating process and how we model the data. Finally we discuss why it was necessary to also explore the problem in a higher dimensional setting and the extended data generating process, what those data look like and how we modelled them.

Chapter 4 discusses policies in general: What they are and how we structure them. We explain the difference between static and adaptive policies and show examples of both in addition to explaining how we apply the policies to our data in order to yield decisions.

Chapter 5 demonstrates how the policies were optimized. We give a summary overview of the optimization engine applying Stochastic Gradient Descent and explaining how we modify it to solve our problem. We also give a thorough explanation of the different metrics we optimize our policies for and how we iterate over and finalize the policies.

Chapter 6 provides the details of the experiments we performed to test the hypoteses. It has a thorough description of how the experiments were conducted in order to ensure robustness of the conclusions and what the results were measured against. We also go into a deeper explanation of the policies we used during experimentaion. Finally, we present the results obtained from the experiments.

Chapter 7 contains a discussion of our results and what we need to consider when interpreting them. We conclude the thesis with a discussion about what we might have done differently and how that would affect the results and their implications before we suggest directions for building upon this work.

Appendix A features a more thorough explanation of some concepts and details about the implementation.

Appendix B shows some alternative settings that were explored and that might be interesting approaches to improve on in further work.

1.5 Definitions and Symbols

Symbol	Meaning
π	Policy
Θ	A Bayesian model of the world with assumed interactions between variables and a distribution over the effects of each interaction
θ	A Belief about the model Θ containing single point values of each interaction effect in the model. The values of the believed effects could be randomly sampled from the distribution, or they could be the exepcted value of the distribution.
U	Utility, a measure of the usefulness of an action-outcome pair to the decision maker
U_t	The utility at time t
R	Reward, a measure of the usefulness of an outcome to the decision maker
$\mathcal{C}$	Cost, the cost of an action for the decision maker
a	An action
A	The set of possible actions (in this thesis, always 2)
|A|	The number of possible actions
$\mathbf{a}$	A vector containing actions $a_i \in A \quad \forall i \in [1, N]$ for multiple individuals
x	The value of one feature in one individual
X	The set of all possible values a feature can hold
|X|	The number of possible values a feature can hold
$\mathbf{x}$	A vector containing multiple feature values for an individual
$\mathbf{X}$	A matrix containing rows of $\mathbf{x}_i \quad \forall i \in [1, N]$ for multiple individuals
N	Number of individuals in consideration
y	An outcome
Y	The set of possible values an outcome can take
|Y|	The number of possible outcomes
$\mathbf{y}$	A vector containing outcomes $y_i \in Y \quad \forall i \in [1, N]$ for multiple individuals
z	A sensitive group identifier
Z	The set of possible sensitive groups
\|Z\| | The number of sensitive groups (in this thesis, always 2) |

Symbol	Meaning
$\mathbf{z}$	A vector containing sensitive group belonging $z_i \in Z \quad \forall i \in [1, N]$ for multiple individuals
F	Fairness, a metric defining how fair a set of actions and corresponding outcomes. Perfect fairness is 1
$\bar{F}$	Fairness violation, a metric defining the violation of fairness of a set of actions and corresponding outcomes. Perfect fairness violation is 0
B	Balance, a metric defining the violation of the balance criterion for fairness of a set of actions and corresponding outcomes. Perfect balance is 0
C	Calibration, a metric defining the violation of the calibration criterion for fairness of a set of actions and corresponding outcomes. Perfect calibration is 0
γ	The weighting of balance versus calibration in the final fairness metric
FU, U_F	Fair utility, a weighted average of fairness and utility wherein the decision maker determines how much emphasis to put on own utility versus fairness for the objects of the decisions
λ	The weighting of fairness compared with utility in the final utility
P_θ	The probability of an event under a certain belief, equivalent to $p(\dots\|\theta)$
δ	Discounting factor for utility, fairness and fair utility, reducing the weight placed on future utility, fairness violation and consequently fair utility
$D(t)$	A generalized discount formula for a time t timepoints into the future
$\mathbf{W}$	The matrix making up the weights of a policys computational layer
w_{ij}	A single weight in $\mathbf{W}$
LF	Latent Feature, a feature used in the data generating process that is unobservable
ω	Mapping of discrete input values from x to a vector multipliable by $\boldsymbol{W}$
Ω	A vector produced by an ω consuming an x
ϕ	The Probability Density Function of the Normal distribution
$\mathcal{D}$	Dataset
${}_{T=t}\mathcal{D}_a^i$	The value of a, the action, of the i-th individual of the dataset seen at time $T = t$.
d	Number of features for an individual: $\|\boldsymbol{x}\|$

Table 1.1: Definitions and Symbols

CHAPTER 2

The Problem

The problem that we are trying to solve has a number of elements to it. We are trying to make decisions that are both useful for the decision maker and fair. We are also trying to optimize these decisions not only with regard to to the outcome of the current set of decisions, but also with regard to all future decisions.

While it may appear difficult to optimize current decisions with regard to the outcome from future decisions, this is simply a trade-off between taking decisions for optimal outcomes in the current set of decisions versus taking the decisions that lead to maximum increase in actionable information for optimizing future decisions. Thus we need to weigh the reward from the outcomes of the current decisions with the information gathered from the outcomes of those same decisions. As a result we may find it better to take decisions at the present if those decisions are likely to yield information that enables us to take better decisions in the future whose expected improvement is larger than the detriment caused to our current decisions. This trade-off between current and future reward is, in the field of automated decision making, commonly referred to as the exploration–exploitation dilemma(Sutton and Barto 2020, p. 3).

In order to measure the quality of our decisions we need to define a few concepts:

- *Utility:* This is the utility of the outcome from a decision for the decision maker. This will need to be measured by the decision maker.
- *Fairness:* As we are trying to make decisions that are both useful *and* fair we need to define what we mean by fair.
- *Future discount:* It may be that we would like to put more emphasis on the present outcomes than the future outcomes. This could be because we prefer something good at the present and would want more future improvement to sacrifice current utility. It could also be that the future utility is uncertain as it is only an estimate based on longer-term predictions than the current utility, and that we want some discount for our willingness to tackle that increased risk.
- *Utility vs fairness:* We need to define how we weigh utility against fairness. At some point our decision maker will have to choose between a policy

with a higher utility and one with lower fairness violation. In order to make this choice we need to define how much we care about one versus the other.

2.1 The Setting

We could have chosen almost any setting where decisions are to be made that have an impact on individuals. The main point is that we need to explore/exploit and we need to find the optimal trade-off as well as the optimal decision with regard to fairness and utility. Thus the requirements for a setting we can use are that it needs to be a setting where a decision maker gets a reward from the outcomes of the actions chosen, we must be able to discuss the fairness of the decisions taken and we need to gain information from the outcomes of the choices made in order to be able to have an interesting problem of balancing present and future rewards.

In order to fulfill these requirements we chose the setting of admissions to tertiary schooling. We then have information from secondary school graduation to base admissions on. We will need to be able to group the students into two groups (though we don't necessarily need to define what those groups are) in order to be able to talk about fairness. Finally we will have outcomes in the form of graduation grades from tertiary education determining the reward of our decision maker, namely the college or university in question.

2.2 Utility

The utility of the institution will depend on the cost, $\mathcal{C}$, of admitting a student and the reward, R, of having a student graduate. Probably the reward of a student graduating is higher the better the graduation result, y. Thus we might set a formula such as

$$U(a, y) = \begin{cases} R(y) - \mathcal{C} & \text{if a='admit'} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

to give the value of each student admitted. Now at time of admission we don't know y, so calculating this utility is difficult. We could use 2.1 on the students that we already have observed and consider representative of our current batch of students or we can calculate an expected utility of the student by using the distribution $P(y|\mathbf{x})$, where $\mathbf{x}$ is the information about a student known to the decision maker, in

$$E[U|\mathbf{x}] = \begin{cases} \sum_{Y} R(y)P(y|\mathbf{x}) - \mathcal{C} & \text{if a='admit'} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

This requires a way to estimate the distribution $P(y|\mathbf{x})$.

Now, if we are given a policy π, that is a function assigning a probability distributions over the set of possible actions for every possible input $\mathbf{x}$ (see

Chapter 4 for a formal definition), then we can include it in the conditionals and calculate the expected utility in the following way:

$$E[U|\mathbf{x},\pi] = (\sum_{Y} R(y)P(y|\mathbf{x}) - \mathcal{C})\pi(a = \text{'admit'}|\mathbf{x}) + 0 \cdot \pi(a = \text{'not admit'}|\mathbf{x})$$

$$E[U|\mathbf{x},\pi] = \pi(a = \text{'admit'}|\mathbf{x}) \sum_{Y} R(y)P(y|\mathbf{x}) - \mathcal{C} \tag{2.3}$$

The problem with this approach is that we need a set of students to calculate a utility, and thus we can't define the policy before the students have applied. We can, however, note that the expected utility of a student depends only on the student's results from secondary education,$\mathbf{x}$, which means we use the probability of obtaining an applicant with input $\mathbf{x}$ to calculate a new expected utility, this time before seeing any applicants.

$$E[U|\pi] = \sum_{X} P(\mathbf{x})\pi(a = \text{'admit'}|\mathbf{x}) \sum_{Y} R(y)P(y|\mathbf{x}) - \mathcal{C}$$

This could be written more generally as

$$E[U|\pi] = \sum_{X} P(\mathbf{x}) \sum_{A} \pi(a|\mathbf{x}) \sum_{Y} U(a,y)P(y|\mathbf{x})$$

where $U(a,y)$ is 2.1 or even shorter as

$$E[U|\pi] = \sum_{X,Y,A} \pi(a|\mathbf{x})U(a,y)P(y,\mathbf{x}) \tag{2.4}$$

In the case where the number of possible combinations of all $\mathbf{x}$, y and a make Equation (2.4), intractable a Monte Carlo approximation will be required

$$E[U|\pi] = \frac{1}{N} \sum_{\substack{i=0 \\ \mathbf{x}_i,y_i \sim P(y,\mathbf{x})}}^{N} \sum_{a \in A} \pi(a|\mathbf{x}_i)U(a,y_i) \tag{2.5}$$

2.3 Fairness

For group fairness we have two reasonable measures of fairness, *balance* and *calibration*(Definition 1.2.2). Since it is impossible to satisfy both(Kleinberg, Mullainathan, and Raghavan 2016) we have to either choose one, or to optimize over a weighted average of the two. The latter approach seems more reasonable, as it would be better to have a small violation of both measures than no violation of one of them and a large violation on the other one. Thus our fairness function can be

$$\bar{F} = \gamma B + (1 - \gamma)C \tag{2.6}$$

Where $\bar{F}$ is total fairness violation, B is violation of the fairness criterion *balance* given in Equation (1.2), C is violation of the fairness criterion *calibration* given in Equation (1.1), and γ is a constant determining the weighting between the two.

These measures of fairness are easy to calculate on a dataset where we have actions and outcomes, but we want to apply them on our applicants, for whom we do not have any actions yet, and we don't know what the outcomes will be. Thus we need, as in the case of utility, to calculate the expected fairness violation.

Expected Fairness

Dimitrakakis et al. (2017) uses a formula for expected balanced fairness violation of a policy given a model:

$$f(\pi) = \int_{\Theta} \sum_{a,y,z} \left| \sum_{x} \pi(a|x) \left[P_\theta(x, z|y) - P_\theta(x|y) P_\theta(z|y) \right] \right|^P d\beta(\theta) \tag{2.7}$$

This can be written in a more general form as

$$f(\pi) = \int_{\theta \in \Theta} \beta(\theta) \int_{a \in A} \int_{y \in Y} \int_{z \in Z} E[\bar{F}(a, y, z)|\theta, \pi] dz dy da d\theta \tag{2.8}$$

where $\bar{F}(a, y, z)$ is a function defining the violation of fairness in a dataset, in this case the balanced fairness criterion as defined in Equation (1.2). The violation is defined as the divergence between the two distributions

$$\bar{F}(a, y, z) = D(P(a, z|y) \| P(a|y) P(z|y))$$

In this case the divergence measure is a variant of Total Variation Distance

$$D(P(a, z|y) \| P(a|y) P(z|y)) = |P(a, z|y) - P(a|y) P(z|y)|^P$$

In the case where the model Θ, a, x, y, and z are discrete, the integrals above are reduced to sums, and, as long as there are only a few possible values each variable can take, it is computationally tractable to compute the expectation as described in Equation (2.7). In all of the cases that we will explore it is only the Θ that is continous, and the a-, y- and z variables make a manageable number of combinations. In this case we will rewrite Equation (2.8) as

$$f(\pi) = \sum_{\theta \sim \Theta} \sum_{a,y,z} E[F(a, y, z)|\theta, \pi] \tag{2.9}$$

and in one of our cases the x variable takes on a sufficiently low cardinality that we can use as Dimitrakakis et al. 2017

$$E[\bar{F}(a, y, z)|\theta, \pi] = \left| \sum_{x} \pi(a|x) \left[P_\theta(x, z|y) - P_\theta(x|y) P_\theta(z|y) \right] \right|^P \tag{2.10}$$

to calculate balance. This decomposition of expected balance given π, Θ is only computationally tractable for the balance criterion and not the calibration criterion.

In a higher cardinality setting on the x, where x is continous instead of discrete, or for the calibration criterion, Equation (2.10) becomes intractable as a sum and we will use a Monte Carlo approximation of $E[\bar{F}(a,y,z)|\theta,\pi]$

$$E[\bar{F}_B(a,y,z)|\theta,\pi] \approx \left|\hat{P}(a,z|y,\theta,\pi) - \hat{P}(a|y,\theta,\pi)\hat{P}(z|y,\theta,\pi)\right|^P \tag{2.11}$$

$$E[\bar{F}_C(a,y,z)|\theta,\pi] \approx \left|\hat{P}(y,z|a,\theta,\pi) - \hat{P}(y|a,\theta,\pi)\hat{P}(z|a,\theta,\pi)\right|^P \tag{2.12}$$

where $\hat{P}(a,z|y,\theta,\pi)$, $\hat{P}(a|y,\theta,\pi)$, $\hat{P}(z|y,\theta,\pi)$, $\hat{P}(y,z|a,\theta,\pi)$, $\hat{P}(y|a,\theta,\pi)$ and $\hat{P}(z|a,\theta,\pi)$ are approximated according to Algorithm 1.

Algorithm 1: Approximating fairness probability distributions from simulated data

begin

Simulate a dataset $\mathcal{D}$ from θ by sampling $x,y,z \sim P(x,y,z|\theta)$

Set $\hat{P}(Y=y,Z=z|\theta) = \frac{\sum_{\mathcal{D}^i\in\mathcal{D}} I(\mathcal{D}^i_y=y,\mathcal{D}^i_z=z)}{|\mathcal{D}|} \quad \forall y \in Y \quad \forall z \in Z$

Set $\hat{P}(A=a|Y=y,Z=z,\theta,\pi) = \frac{\sum_{\mathcal{D}^i\in\mathcal{D}} I(\mathcal{D}^i_y=y,\mathcal{D}^i_z=z)\pi(A=a|\mathcal{D}^i_{\mathbf{x}})}{\sum_{\mathcal{D}^i\in\mathcal{D}} I(\mathcal{D}^i_y=y,\mathcal{D}^i_z=z)} \quad \forall y \in Y \quad \forall z \in Z \quad \forall a \in A$

Set $\hat{P}(A=a|\theta,\pi) = \frac{\sum_{\mathcal{D}^i\in\mathcal{D}} \pi(A=a|\mathcal{D}^i_{\mathbf{x}})}{|\mathcal{D}|} \quad \forall a \in A$

Set $\hat{P}(Y=y,Z=z|A=a,\theta,\pi) = \frac{\hat{P}(Y=y,Z=z,\theta)\hat{P}(A=a|Y=y,Z=z,\theta,\pi)}{\hat{P}(A=a,\theta,\pi)} \quad \forall y \in Y \quad \forall z \in Z \quad \forall a \in A$

Estimate $\hat{P}(a,z|y,\theta,\pi)$, $\hat{P}(a|y,\theta,\pi)$, $\hat{P}(z|y,\theta,\pi)$, $\hat{P}(y|a,\theta,\pi)$ and $\hat{P}(z|a,\theta,\pi)$ similarly as $\hat{P}(y,z|a,\theta,\pi)$

For the balance criterion this is just a computationally more efficient form of, in the notation of Dimitrakakis et al. 2017,

$$E[\bar{F}(a,y,z)|\theta,\pi] \approx \left|\sum_{x\sim P(x|a,y,z,\Theta,\pi)} P_\theta(x,z|y) - P_\theta(x|y)P_\theta(z|y)\right|^P$$

2.4 Future discount

When computing and comparing utility over time we need to make the sum of present and future utility finite. There are two ways to do this:

We can compute the utility for each timepoint for a finite horizon T

$$U = \sum_{t=0}^{T} U_t \tag{2.13}$$

or we can compute utility for an infinite horizon, but using discounting to downweigh future utility over current. For example

$$U = \sum_{t=0}^{\infty} U_t \cdot \delta^t \qquad 0 < \delta < 1 \tag{2.14}$$

The practical difference between the two is that the former has a piecewise discount function whereas the latter has a continous discount function. Thus they can both be written as

$$U = \sum_{t=0}^{\infty} U_t \cdot D(t) \tag{2.15}$$

where in the case of 2.13 we get

$$D(t) = \begin{cases} 1 & t <= T \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

and in the case of 2.14 we get

$$D(t) = \delta^t \tag{2.17}$$

In practice, however, it will be impossible to compute an infinite sum of utilities that have to be obtained through experiments. Thus they will both need to be calculated as

$$U = \sum_{t=0}^{T} U_t \cdot D(t) \tag{2.18}$$

With δ sufficiently small or T sufficiently large, the distinction between 2.15 and 2.18 will be minimal.

We will start without a future discount on fairness and utility. This means that we value future utility as high as present utility, and we will not favour present fairness over future fairness. Our goal here is to derive an algorithm that will be capable of taking better decisions over time than a greedy algorithm that would simply optimize for the present utility and fairness. If we manage to demonstrate that it does in fact deliver better decisions over time without any discount on future utility and fairness we can consider adding a discount to make the problem more diffficult.

2.5 Utility versus fairness weighting

In order to find the optimal decision we need to define how we measure optimality. This we will do as a weighted average between the utility of the institution and a fairness measure similarly to how we weighted balance and calibration. Instead of using the *fairness violation* as we have discussed how to measure, which is optimal when it is 0, we will define a fairness metric using the normalization

procedure from Section 2.5. This procedure ensures that both utility and fairness violation are on the same scale, and are more or less within $[0, 1]$, but where utility is optimal at 1 and minimal at 0 and fairness violation is optimal at 0 and minimal at 1. Instead we define fairness as the absence of fairness violation.

$$F = 1 - \bar{F} \tag{2.19}$$

This gives us fairness that is optimal at 1 and minimal at 0 as utility.

Then we can define fair utility as the weighted average of fairness and utility

$$U_F = \lambda U + (1 - \lambda)F \tag{2.20}$$

which is similar to the definition used by Dimitrakakis et al. (2017), but adding instead of subtracting the fairness component.[1]

Finding a suitable weight

One problem with weighing utility and fairness is that utility and fairness are not measured in the same units or on the same scale. Thus setting $\lambda = 0.5$ does not imply equal weighting of fairness and utility. The cleanest way of making an informed weighing between the two would be to either rescale utility and fairness to have the same range, or to rescale λ such that $\lambda^* = \lambda \frac{range(\bar{F})}{range(U)}$.

Calculating the ranges, however, is not trivial. First of all there is a difference between the theoretical range and the range we might expect them to have. Secondly, the calculation of the maximum theoretical range is not necessarily straightforward either.

For the utility it is quite easy to calculate the maximum theoretical range given some parameters of the utility function.

$$U_{min} = \frac{1}{N} \sum_{i=1}^{N} U(a = 1, y = 0)$$

$$U_{max} = \frac{1}{N} \sum_{i=1}^{N} U(a = 1, y = max(y))$$

This, however, is not a realistic range, because regardless of the policy we cannot expect such datasets.

[1] Defining $U_F = (1 - \lambda)U - \lambda F$ as our objective function leads to U_F decreasing with increasing λ. This has two unfortunate consequences. First it gives the appearance of less useful policies with higher weighting of fairness. Even though it is not true, that appearence is unfortunate. Secondly it makes it more difficult to judge the perfomance across λs. In addition it requires us to use the tongue-twister "deviation from fairness" when talking about the fairness metric. The new definition makes fairness a metric we want to be higher, and it makes the weighted average using λ more sensible in that you either try to maximize utility, or fairness, or a combination, making U_F more or less constant regardless of λ.

Similarly for Fairness, the fairness part at the core of the formula, $P_\theta(x, z|y) - P_\theta(x|y)P_\theta(z|y)$, has a theoretical maximum of 1 as they are all probabilities. The entire formula, however, is a sum over all relevant variables, and this core cannot be 1 for all combinations of variables. Being a sum its range is also very much dependent on the number of possible values of a, x, y and z.

The solution we use here is to train a policy on a huge dataset using both $\lambda = 0$ and $\lambda = 1$. We then set

$$U_{min} = max(U_{\lambda=0}, U_{a=\mathbf{0}})$$

$$U_{max} = U_{\lambda=1}$$

$$\bar{F}_{min} = 0$$

$$\bar{F}_{max} = F_{\lambda=1}$$

We limit the lower value on the range of U because the policy trained with $\lambda = 0$ will give no regard whatsoever to utility and can be wildly variable in utility since any policy satisfying $\pi(a = 1|x, \theta) = p \quad \forall p \in [0, 1]$ will give perfect fairness, while the utility of the policy $\pi(a = 1|x, \theta) = 1$ could be very negative. Thus we limit how low the policy is allowed to pull the lower bound on the range to the utility of $\pi(a = 1|x, \theta) = 0$. This gives us

$$U \in [U_{min}, U_{max}]$$
$$\bar{F} \in [\bar{F}_{min}, \bar{F}_{max}]$$

While these ranges are not absolute, in that it is perfectly possible to measure either metric outside of their ranges, it gives us a way to compare their magnitude and range. Using these ranges we can then normalize both before weighing them

$$U' = \frac{U - U_{min}}{U_{max} - U_{min}}$$
$$\bar{F}' = \frac{\bar{F} - \bar{F}_{min}}{\bar{F}_{max} - \bar{F}_{min}}$$

Giving us

$$U', \bar{F}' \in [0, 1]$$

CHAPTER 3

The Setting

This chapter contains first a discussion of the choice of problem setting, then a discussion about how the data generating process was designed as well as some graphs, algoritms and plots descrbing the process and the resulting dataset. Finally, the model that seeks to model the data-generating process is presented, with a discussion about the assumptions underlying the model and a derivation of the closed-form solutions to the updates of the model to new posterior distributions over the parameters.

We could have chosen almost any setting where decisions are to be made that have an impact on individuals. The main point is that we need to explore/exploit and we need to find the optimal trade-off as well as the optimal decision with regard to the uncertainty of the model, the parameters and fairness and utility. This is considerably easier in a setting where we generate our own data, giving the ability to generate an arbitrary amount of data as needed.

As we are generating our own data we could have chosen any setting, and settled on school admissions. This consists of a set of applicants with a set of known features which is then fed into a policy taking actions, to admit or not, and then for those applicants that are admitted we receive an outcome.

We started with a setting in which we had three features in the dataset.

- A protected variable (z in Definition 1.2.2) that the policy is not allowed to take into account and that we use to determine the fairness of the policy throughout Chapter 2. It is dichotomous with almost, but not quite, 50/50-split in the applicant pool. It is not specified what this feature is, but with the ratio gender is one possible interpretation.

- A discrete feature $x \in [0, 4]$ that represents the graduation grade of the applicant from secondary eduaction. This is the input we give to the policy that is our decision-maker as described in Definition 4.0.1.

- A discrete feature $y \in [0, 4]$ that represents the graduation grade of the applicant, if admitted, from our tertiary education institution. This is the outcome we use in Definition 1.2.2 and that we use throughout Chapter 2 to compute the fairness and utility of a set of decisions.

In order to test the scalability of the algorithm we also apply a second, more complex, setting. It is equal to the first in the protected variable z and the outcome y, but has more dimensions and higher cardinality in the input, $\mathbf{x}$, that the policy uses to make its decisions.

3.1 Low-dimensional setting

Generating the data

The idea behind the generated data in the simple setting is that people have different academic potential. Then they are able to live up to this potential to different degrees in secondary education. Some underperform and others overperform. And this performance relative to potential might not be equally distributed between the groups even if the potential is. This could be students with rich parents being able to hire private tutors that students with poor parents cannot if the groups were created based on economic background. It could be girls being better suited to how school is set up than boys. It could be immigrant students forced to learn both the school curriculum *and* a new language, or simply having to cope with a new and strange environment. The students then go through tertiary education and we assumed then that the advantages or disadvantages in primary and secondary education would fade, but that better results from secondary education implies higher knowledge, and thus an advantage in tertiary education.

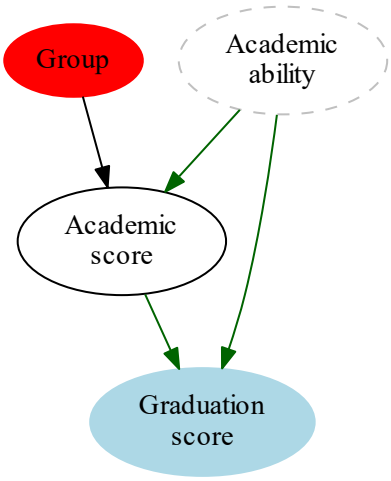


Figure 3.1: Graph of variables in use for the data generation. Red is the sensitive variable, Blue is the target variable. The dashed variable is latent. Green arrows are positive influences while red would be negative.

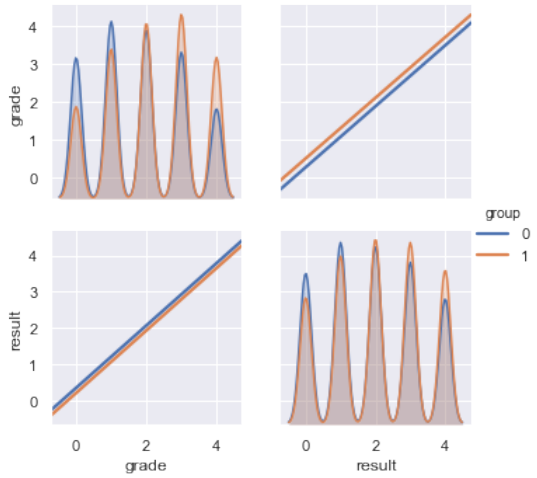


Figure 3.2: Pairplot of a set of 1000 generated data points grouped by the sensitive variable. Distributions of the variable are on the diagonal while pairwise correlations between the variables are on the off-diagonal.

The data was generated according to Algorithm 2:

Algorithm 2: Generating low-dimensional data

begin
  for $i \in 1 : N$ **do**
    Sample $Z_i \sim p(Z)$
    Sample $ability_i \sim N(\mu_A(Z_i), \sigma_A(Z_i))$
    Sample $X_i \sim N(ability_i + advantage(Z_i), \sigma_X)$
    Set $E[Y_i] = (1 - \lambda)ability_i + \lambda X_i$
    Sample $Y_i \sim N(E[Y_i], \sigma_Y)$
    Discretisize X and Y into, in expectation, almost equal batches with some more in the middle batches

Modelling the data

The goal of our model is to predict the probability distribution for graduation grades given admission grades and group belonging. In this we could simply use a Dirichlet-Multinomial model and model, for each x and x, the probability distribution of y. Multinomial is probably not the best in this case, though, seeing how the five categories of output variables are connected (4 is closer to 5 than 1 is, it's very unlikely that the probability of getting a 5 or a 1 are high while 2, 3, and 4 are low).

Another approach is to use a binomial distribution with $n = |Y| - 1$ and $p_{x,z} \in [0, 1]$ as $p(Y = y|X = x, Z = z)$. We could view each graduation score as the number of correct answers to a $|Y| - 1$-question test where the probability of answering correctly is $p_{x,z}$, with y being the number of correct answers.

The advantage (or disadvantage) of the Beta-Binomial model is making a connection between the probabilities of neighbouring scores. Thus if you enter tertiary education with a 0 there is a lower probability of you graduating with a 4 than with a 3, which again has a lower probability than a 2. With large datasets this structure will come in both models, but a stricter model when we can be fairly confident about the validity of the constraints posed makes our model more robust when facing less data.

Beta-Binomial model

$$y \sim Bin(n, p|x, z) \qquad n = |Y| - 1$$
$$p|x, z \sim Beta(\alpha^y_{xz}, \beta^y_{xz})$$

When updating α^y_{xz} and β^y_{xz} we have regular Beta-Binomial update with ${\alpha^y_{xz}}^{(t)} = {\alpha^y_{xz}}^{(t-1)} + \sum_{X=x,Z=z} y$ and ${\beta^y_{xz}}^{(t)} = {\beta^y_{xz}}^{(t-1)} + \sum_{X=x,Z=z} n - y$.

For $p(X = x|Z = z)$, viewing x as the number of correctly answered questions in a $(|X| - 1)$-question test where the probability of answering correctly, p_z, depends on which group the student belongs to would not give the best result as it might be that the distribution of different values for x don't follow a Binomial distribution for each z.

$$x \sim Dirichlet(\boldsymbol{\alpha^x})$$

When updating α^x_z we have a regular Dirichlet-Multinomial update with ${\alpha^x_z}^{(t)} = {\alpha^x_z}^{(t-1)} + \sum_{X=x,Z=z} x$.

This model has the following priors on the Beta-distributions:

$$\alpha^y, \beta^y \in \mathbb{R}^{|X|\times|Z|}$$

$$\alpha^x, \beta^x \in \mathbb{R}^{|Z|}$$

Now we could quite reasonably set fairly informative priors on several of these. It's difficult to defend a distinction between the two groups on the priors, but we could expect a difference in α^y and β^y for different values of x so that we might choose as our priors higher $\alpha^y_{x\cdot}$ for higher x and conversely for $\beta^y_{x\cdot}$. Another option is to set an uninformative prior of 1 on all.

3.2 Extended setting - higher dimensional input

In order to make this problem a bit harder we will increase the dimensionality with some features that we can expect will be less directly correlated with our target, the graduation grade of the student. According to United States District Court 2019, pp. 18-22, Harvard uses four profile features in their admissions process, each of which is judged on a scale 1-4 with 1 being the best and 4 being the weakest. In addition + and - is available for each rating, giving us 12 discrete levels[1]. In addition to the four profile features there are three school support

[1] 5 and 6 are also available and indicate wekaness or special circumstances giving us potentially up to 18 rating levels for each feature, but we can stick to 1+-4-. Considering the approach in modelling this with a Bayesian Logistic Regression it doesn't really matter from a computational perspective how many discrete values the features can take. In fact they could just as well be continous.

features in the same range, indicating the strength of the recommendations the candidates get. For our purposes we could collapse these into one, giving us the following features:

1. *Academic* - Reflecting the applicant's academic strength based on grades, tests, recommendations, academic work and strength of high school.
2. *Extracurricular* - Reflecting extracurricular activities, and the potential for those at Harvard, and may also take into account that a student may not have had the opportunity to engage in such activites.
3. *Athletic* - Reflecting the activity level and the skill level of the student in athletic pursuits.
4. *Personal* - An assessment about what contribution a student might give to the school community.
5. *Recommendations* - An indication of the strength of the recommendations from the student's high school.

This gives us five features per candidate instead of one in addition to the protected group feature z. This means that we need to create a new data-generating process that produces for each candidate another 4 features. We also need to update our model. Our simple Beta-Binomials models no longer work and we need to consider how to deal with interactions and covariation between the features. Lastly we will need to update our policies to be able to take into account additional features. The policy is not something we need to change dramatically or think much about, it just needs to be able to handle additional features.

Generating the data

Our original data-generating process was built on the idea that each person has an underlying academic ability and that their grades are a reflection of this ability, but with some random noise and some systematic bias between groups. Extending this with our extra features means devising mechanisms for the connection between each of our features and academic ability. These connections are much less obvious to identify and quantify, and any such connection used will be far more controversial. However, it is necessary to set them in order to get some data.

Originally we had a latent variable we called "ability" (which perhaps should be called "academic ability") which influenced academic performance. It was evenly distributed between our sensitive groups. Now we will make another latent variable we will call "energy" which describes how energetic a person is, which will positively influence both academic (along with academic ability) performance, extracurricular performance and athletic performance. In addition we will add a differentiation of the latter two among the sensitive groups as we did for academic performance. The Personal feature we will make truly random and let be simply a noise feature while the recommendation feature

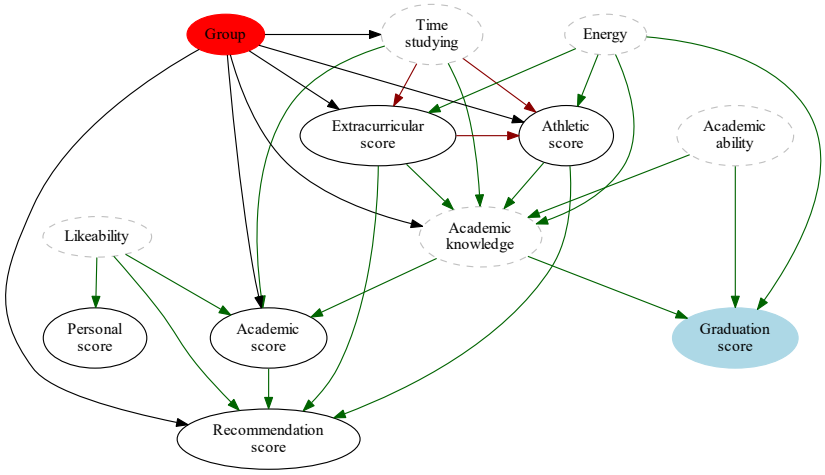


Figure 3.3: Graph of variables in use for the extended data generation. Red is the sensitive variable, Blue is the target variable. Dashed variables are latent. Green arrows are positive influences while red are negative.

will be based with some weighting on all the other features, plus a bias between the groups. In addition to all this it is tempting to make some interactions between the variables in order to make sure that modelling the process will be hard and simple linear models will be imperfect. Some such interactions could be a positive effect on academic performance from extracurricular and athletic activities because you will learn things also there that might improve your academic learning as well as a negative effect due to time spent on those other activites instead of spending that time on academic pursuits. It might also be possible to include a positive effect of personality on academic results in order to capture a possible effect of your teachers giving you better grades if you are more likeable. This effect should then probably be negated in the graduation score because it is a part of the academic score that doesn't actually reflect ability. These last interactions might make it more difficult for the model to learn and predict, which is a good thing, but it is also quite possible that their effects will in any case drown in random noise and the lack of knowledge of our latent variables.

We can then generate data according to Algorithm 3

Modelling the extended setting

We now must be careful to not let the knowledge we have of the data generating process leak too much into our modelling process. There are two main issues that we now have to handle that we didn't previously. Those are interactions and covariance between the variables.

No longer given that we have a monotonic relationship between the other

Algorithm 3: Generating high-dimensional data

begin
- **for** $i \in 1 : N$ **do**
  - Sample $Z_i \sim p(Z)$
  - **for** $LF \in \{Likeability, Time, Energy, Ability\}$ **do**
    - Sample $LF_i \sim p(LF|Z_i)$
  - **for** $X \in \{Extracurricular, Athletic, Personal\}$ **do**
    - Sample $X_i \sim p(X|\boldsymbol{LF_i}, Z_i)$
  - Sample $Knowledge_i \sim p(Knowledge|\boldsymbol{LF}_i, \boldsymbol{X}_i, Z_i)$ into $\boldsymbol{LF}_i$
  - Sample $Academic_i \sim p(Academic|\boldsymbol{LF}_i, Z_i)$ into $\boldsymbol{X}_i$
  - Sample $Recommendation_i \sim p(Recommendation|\boldsymbol{LF}_i, \boldsymbol{X}_i, Z_i)$ into $\boldsymbol{X}_i$
  - Sample $Y_i \sim p(Y|\boldsymbol{LF}_i)$
- Discretisize $\boldsymbol{X}$ and $\boldsymbol{Y}$ into, in expectation, almost equal batches with some more in the middle batches

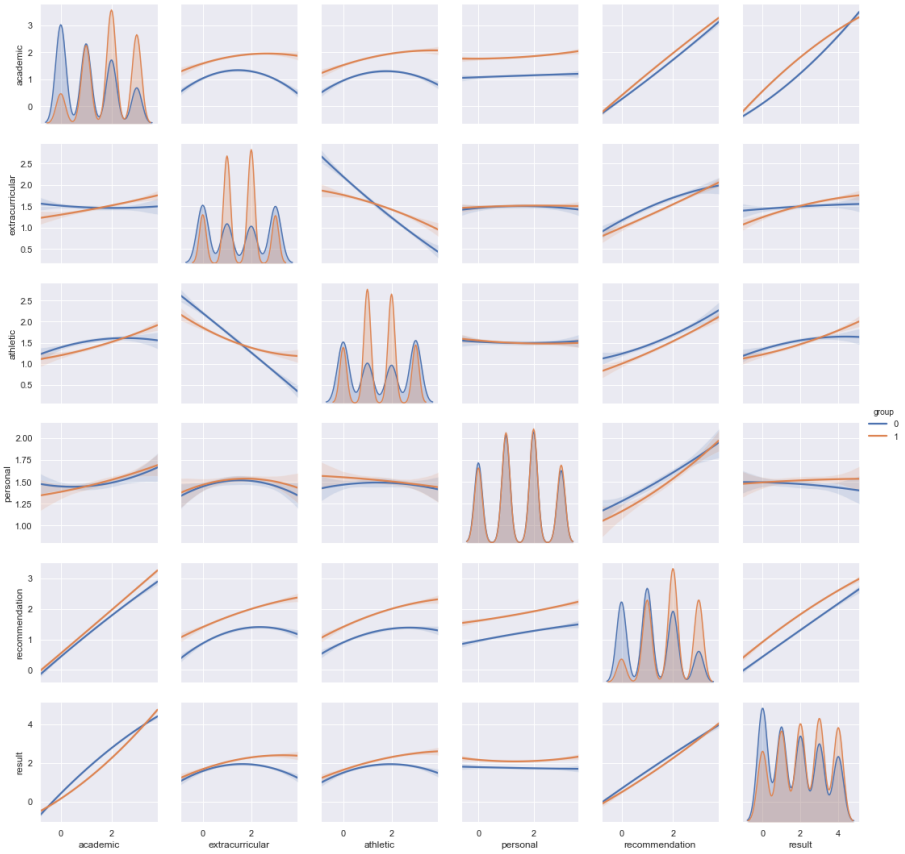


Figure 3.4: Pairplot of a set of 1000 generated data points grouped by the sensitive variable. Distributions of the variable are on the diagonal while pairwise correlations between the variables are on the off-diagonal.

features and our graduation grade, such as we could assume for our academic result feature. Thus perhaps some Dirichlet is more appropriate. Ideally we should use some more advanced modelling methods, but for computational purposes we need updates that have closed form solutions. This constraint applies primarily to our testing because in order to get good testing results we will need to run this process many times. In a real application of this procedure the computational challenges would be less relevant as we would only deal with one optimization rather than hundreds or thousands.

As before our goal is to predict the probability distribution for graduation grades given our features and the group belonging. Our previous model contained a model for $p(Z = z)$, the probability that an individual is a member of group z, a model for $X|Z = z \sim Bin(|X|, p(z))$ where $p(z) \sim Beta(\alpha_z, \beta_z)$, and a model for $Y|X = x, Z = z \sim Bin(|Y|, p(x|z))$ where $p(x|z) \sim Beta(\alpha_{x,z}, \beta_{x,z})$.

In order to make this as simple as possible without adding too many constraints, one approach is to use the same model and extend it with some more X-values.

The Model

We will number our features as follows: Academic: 1, Extracurricular: 2, Athletic: 3, Personal: 4, Recommendations: 5.

$$Z \sim Beta(\alpha, \beta)$$

Then we don't know how our features are distributed in each group, so it seems sensible to use a Dirichlet-Multinomial distribution. It seems a reasonable assumption that the strength of Recommendations is dependent on academic ability, but for the others it's difficult to assume any correlation. An interesting thing would of course to somehow be able to model these features including their correlations, but we'll leave that for later. This gives us

$$X_i \sim Dirichlet(\boldsymbol{\alpha}_z) \qquad i \in \{1, 2, 3, 4\}$$
$$X_5 \sim Bin(|X_5|, p(x_5|x_1, z)) \qquad p(x_5|x_1, z) \sim Beta(\alpha_{x_1,z}, \beta_{x_1,z})$$

Then Y will be a combination of all five features and the group. It makes sense again to use the features to inform a probability in a binomial distribution. Thus $p(y|\boldsymbol{x}, z)$ is what we seek to model. Modelling this for every combination of $\boldsymbol{x}$ like we did previously is now going to give a very slowly updating model as we have, with each feature taking one of four discrete values, $4^5 * 2 = 2048$ combinations.

This means we might need to create another kind of model entirely. Since it is not necessarily true that the score on the extracurricular, athletic and personal features are linearly or monotonically correlated with our target variable they can't be assigned a single coefficient in a linear model. We could treat the features as categorical variables, which will still give us a manageable number of variables, and the academic and recommenation features will probably be

close enough to linear monotonicity that we can assign them just one coefficient, but we lose all interaction effects unless we include them explicitly, which again would, potentially drastically, increase the dimensionality of our model.

If we choose a simple linear regression, $\boldsymbol{y} = \boldsymbol{\beta}^T \boldsymbol{X} + \epsilon$, we have an issue with our target variable being discrete. While it isn't necessarily a problem during model fitting, it is a problem with prediction. We want to predict y from $\boldsymbol{X}$, but a linear model will predict a continous y. We could of course approximate discrete y by giving each value of y a section of the continous range, but we also need to figure out a different issue:

What we need the model for, is to calculate our expected loss for a policy. To accomplish this we have so far taken the sum over every possible combination of $\boldsymbol{x}$, y, z and a of the probability of taking the action a given the input x times the utility of outcome y with action a times the probability of outcome y given input $\boldsymbol{x}$ and group z. In order to do that in our setting we need to compute $p(y|\boldsymbol{x}, z)$ for every combination of $\boldsymbol{x}, z$. This means firstly that we are not really interested in predicting y, we are interested in $p(y|\boldsymbol{x}, z)$. Secondly it means that we need to find a computationally efficient way of calculating our expected loss. With 2048 combinations of $\boldsymbol{x}$ and z this means, with 5 different y and 2 groups, that we need to compute our sum 20,480 times. However efficiently we calculate $p(y|\boldsymbol{x}, z)$ it is becoming hard to calculate the expected loss in sub-millisecond times, which is what we need to get any reasonable testing results.

The obvious solution to the latter problem is to sample from our 20,480 combinations to get an approximation to the expected loss. The alternative is to seek an analytical solution, but that seems difficult to obtain if even possible, and even if it is possible, will slow down any iteration speed we might have.

The solution to the former problem could be to use a Bayesian Logistic Regression to predict p in a Binomial distribution as we did before. This would require us to transform our target from y through p to η where

$$\boldsymbol{\eta} \sim N(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I})$$

$$\boldsymbol{p} = \frac{e^{\boldsymbol{\eta}}}{1 + e^{\boldsymbol{\eta}}}$$

$$\boldsymbol{y} \sim Bin(|Y|, \boldsymbol{p})$$

We now need to transform our recorded target y to the target η for our Bayesian Linear Regression. Transforming from p to η is simply taking the logit of p, $\eta = \log\left(\frac{p}{1-p}\right)$. When transforming y to p we have more options. One of our main two options is to use the MLE-estimator for p, $\hat{p} = \frac{y}{|Y|}$, but for $y = 0$ and $y = 4$ this will give unrealistic $p = 0$ and $p = 1$. In addition to being unrealistic, logit-transforming $p = 0$ and $p = 1$ to η would result in, in the first case $log(0) \to \infty$, and in the second case division by zero. The other option is to use a Bayes estimator. When using a Beta-distribution as conjugate prior this gives a Bayes estimate for the posterior mode of $\hat{p}_B = \frac{y+\alpha}{|Y|+\alpha+\beta}$. If we

use this option we need to choose a prior and by visually trying to minimize $D(P(y|\mathcal{D})\|P(y|\Theta,\mathcal{D}) \quad |\mathcal{D}| \to \infty$ we set $\alpha = \beta = 0.3$ [2]

If we choose a Bayesian linear regression with a Normal-Normal update on our coefficients we have a closed-form solution to the updates. We can find this in (Fahrmeir, Kneib, and Lang 2009, p.151) for the model

$$\eta|\boldsymbol{\beta}, \sigma^2 \sim N(\boldsymbol{X\beta}, \sigma^2\boldsymbol{I})$$

with prior distributions

$$\boldsymbol{\beta}|\sigma^2 \sim N(\boldsymbol{m}, \sigma^2\boldsymbol{M}) \qquad \sigma^2 \sim \text{Inv-Gamma}(a, b)$$

giving posterior update as

$$\tilde{\boldsymbol{M}} = (\boldsymbol{X^T X} + \boldsymbol{M}^{-1})^{-1}$$

$$\tilde{\boldsymbol{m}} = \tilde{\boldsymbol{M}}(\boldsymbol{M}^{-1}\boldsymbol{m} + \boldsymbol{X^T y})$$

$$\tilde{a} = a + \frac{n}{2}$$

$$\tilde{b} = b + \frac{1}{2}(\boldsymbol{y^T y} + \boldsymbol{m^T M^{-1} m} - \tilde{\boldsymbol{m}}^T\tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{\mu}})$$

Now this notation is a bit more difficult to read than necessary with both lower-case and uppercase M signifying mean vector and $\frac{\Sigma}{\sigma^2}$ reespectively. More conventional notation would be μ for the former and we could use S for the latter. Further, since this is an update equation over time we might improve on $\sim$ as signifying updated values, giving us:

$$\eta|\boldsymbol{\beta}, \sigma^2 \sim N(\boldsymbol{X\beta}, \sigma^2\boldsymbol{I})$$

with prior distributions

$$\boldsymbol{\beta}|\sigma^2 \sim N(\boldsymbol{\mu_t}, \sigma^2\boldsymbol{S}_t) \qquad \sigma^2 \sim \text{Inv-Gamma}(a, b)$$

giving posterior update as

$$\boldsymbol{S}_{t+1} = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{S}_t^{-1})^{-1}$$

[2] The two most obvious options would probably be the uniform prior $\alpha = \beta = 1$ and Jeffrey's prior $\alpha = \beta = \frac{1}{2}$. There are however other considerations to take. We have generated these data, so we know what form their distributions will take. In our model, this prior will have a profound effect on the distribution of the different grades as they essentially control the target probability p of each grade. Thus a high value on the priors will make the target probabilty for a result $y = 4$ for the logistic regression be further from 1 than a lower value. If we then try to simulate new data from our model, the difference between this probability and 1 will have a significant effect on the number of $y = 4$ in our simulated data. The process could easily yield a very different distribution by altering a single parameter in the data generating process. How could we set this right in a way that doesn't require access to a large sample of data beforehand (which we wouldn't have, in reality)? We could figure out a way to have this parameter be learnable to fit the distribution we see in the data. Quick attempts at finding closed-form solutions to this problem stranded and it was considered out of scope for this thesis to spend much time on solving it. In a practical application where there was no access to large amounts of simulated data to set the parameters manually we could solve this problem without having to find a closed-form solution as computational demands would be less heavy and time less critical.

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{S}_{t+1}(\boldsymbol{S_t^{-1}}\boldsymbol{\mu}_t + \boldsymbol{X^T y})$$

$$a_{t+1} = a_t + \frac{n_{t+1}}{2}$$

$$b_{t+1} = b_t + \frac{1}{2}(\boldsymbol{y^T y} + \boldsymbol{\mu_t^T S_t^{-1} \mu_t} - \boldsymbol{\mu}_{t+1}^T \boldsymbol{S}_{t+1}^{-1}\boldsymbol{\mu}_{t+1})$$

This involves inverting the S matrix several times during updating. In order to speed up updating we could reparametrize our model to use $\boldsymbol{\Lambda} = \boldsymbol{S}^{-1}$ giving us

$$\eta|\boldsymbol{\beta}, \sigma^2 \sim N(\boldsymbol{X\beta}, \sigma^2\boldsymbol{I})$$

with prior distributions

$$\boldsymbol{\beta}|\sigma^2 \sim N(\boldsymbol{\mu_t}, \sigma^2\boldsymbol{\Lambda_t^{-1}}) \qquad \sigma^2 \sim \text{Inv-Gamma}(a, b)$$

giving posterior update as

$$\boldsymbol{\Lambda}_{t+1} = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Lambda}_t)$$

$$\boldsymbol{\mu_{t+1}} = \boldsymbol{\Lambda}_{t+1}^{-1}(\boldsymbol{\Lambda}_t\boldsymbol{\mu}_t + \boldsymbol{X^T y})$$

$$a_{t+1} = a_t + \frac{n_{t+1}}{2}$$

$$b_{t+1} = b_t + \frac{1}{2}(\boldsymbol{y^T y} + \boldsymbol{\mu_t^T \Lambda_t \mu_t} - \boldsymbol{\mu}_{t+1}^T \boldsymbol{\Lambda}_{t+1}\boldsymbol{m}_{t+1})$$

The downside of this change is that sampling from $\boldsymbol{\beta}|\sigma^2$ requires inverting $\boldsymbol{\Lambda}$, so which of the last two versions is computationally heavier depends on how often we sample versus how often we update. Perhaps the most efficient way computationally is to update both and use the one we don't have to invert every time. This would yield

$$\eta|\boldsymbol{\beta}, \sigma^2 \sim N(\boldsymbol{X\beta}, \sigma^2\boldsymbol{I})$$

with prior distributions

$$\boldsymbol{\beta}|\sigma^2 \sim N(\boldsymbol{\mu_t}, \sigma^2\boldsymbol{S}_t) \qquad \sigma^2 \sim \text{Inv-Gamma}(a, b) \qquad \boldsymbol{\Lambda}_t = \boldsymbol{S}_t^{-1}$$

giving posterior update as

$$\boldsymbol{\Lambda}_{t+1} = (\boldsymbol{X^T X} + \boldsymbol{\Lambda}_t)$$

$$\boldsymbol{S}_{t+1} = \boldsymbol{\Lambda}_{t+1}^{-1}$$

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{S}_{t+1}(\boldsymbol{\Lambda}_t\boldsymbol{\mu}_t + \boldsymbol{X}^T\boldsymbol{y})$$

$$a_{t+1} = a_t + \frac{n_{t+1}}{2}$$

$$b_{t+1} = b_t + \frac{1}{2}(\boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{\mu}_t^T\boldsymbol{\Lambda}_t\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t+1}^T\boldsymbol{\Lambda}_{t+1}\boldsymbol{\mu}_{t+1})$$

Prior distributions

Jeffrey's prior for $p(\boldsymbol{\beta}, \sigma^2)$ is

$$p(\boldsymbol{\beta}, \sigma^2) = p(\boldsymbol{\beta}|\sigma^2)p(\sigma^2) = 1 \cdot \sigma^{-2}$$

With the inverse gamma distribution

$$p(\sigma^2|a, b) \propto \sigma^{-2(a+1)} e^{-\frac{b}{\sigma^2}}$$

giving us $a = b = 0$ for $p(\sigma^2|a, b) = \sigma^{-2}$.

$p(\boldsymbol{\beta}|\sigma^2) = 1$ is an improper prior on $\beta|\sigma^2$. Though it has a proper posterior as long as $\boldsymbol{X}^T\boldsymbol{X}$ has full rank it is unnecessary to limit ourselves to that. Using a proper Normal prior on $\beta|\sigma^2$ with $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{S}_0 = k\boldsymbol{I}$ will cause our posterior to be proper regardless of the data and is equivalent to a ridge regression where higher k will result in less regularization.

Since we have already departed from Jeffrey's prior by using $\boldsymbol{\beta}|\sigma^2 \sim N(\mathbf{0}, k\boldsymbol{I})$ in order to be able to use our model without data we might as well depart from $p(\sigma^2|a, b) = \sigma^{-2}$ in order to use a, b that allows us to sample from, and take the mean of, Inv-Gamma(a, b). The mean is $\frac{b}{a-1}$ and the distribution is only valid for $a, b > 0$. Thus to be able to get a strictly positive mean we need $b > 0, a > 1$ and to be able to sample from the distribution we need $a > 0, b > 0$. Thus setting $a = 1 + \delta, b = \delta$ where δ is an infintesimally small value we are able to use the model even without data. This change alters the prior on σ^2 to, effectively, $p(\sigma^2|a, b) = \sigma^{-2}e^{-\sigma^{-1}}$. While this is a departure from Jeffrey's prior, the difference will be inconsequential the moment we get data to update the model, and so the only practical effect is to give us a proper prior, allowing us to sample parameters with no data.

We might also here have considered some informative priors. This would be especially relevant on the *academic* feature that we could quite reasonably assume to be positively correlated with our outcome. Using an informative, mainly positive prior on this β would make our model more likeliy to yield better results in low-data settings as long as the assumption underlying the choice of prior is valid.

Simulating new data from model

As for the low-dimensional model, we add functionality for simulating new data from a model belief.

This gives a good way to visualize how well the model works, we can generate a large dataset, train the model on that, simulate data from the model and compare those data with the data we trained the model on.

As we see the distribution of each parameter is close to the original data, while some of the correlations between variables are missing or different. This is to be expected as we put a very comlex set of interactions between our generated variables and omitted the correlations when we modelled the features. While it might be possible to improve the model, we can never expect to have a model

Algorithm 4: Simulating high-dimensional data

begin
for $i \in 1 : N$ **do**
  Simulate $z_i \sim p(z|\theta)$
  for $j \in \{Academic, Extracurricular, Athletic, Personal\}$ **do**
    Sample $X_{ij} \sim p(x|z_i, \theta)$
  Sample
  $Recommendation_i \sim p(Recommendation|Academic_i, z_i, \theta)$
  into $\boldsymbol{X}_i$
  Sample $y_i \sim p(y|\boldsymbol{X}_i, z_i, \theta)$

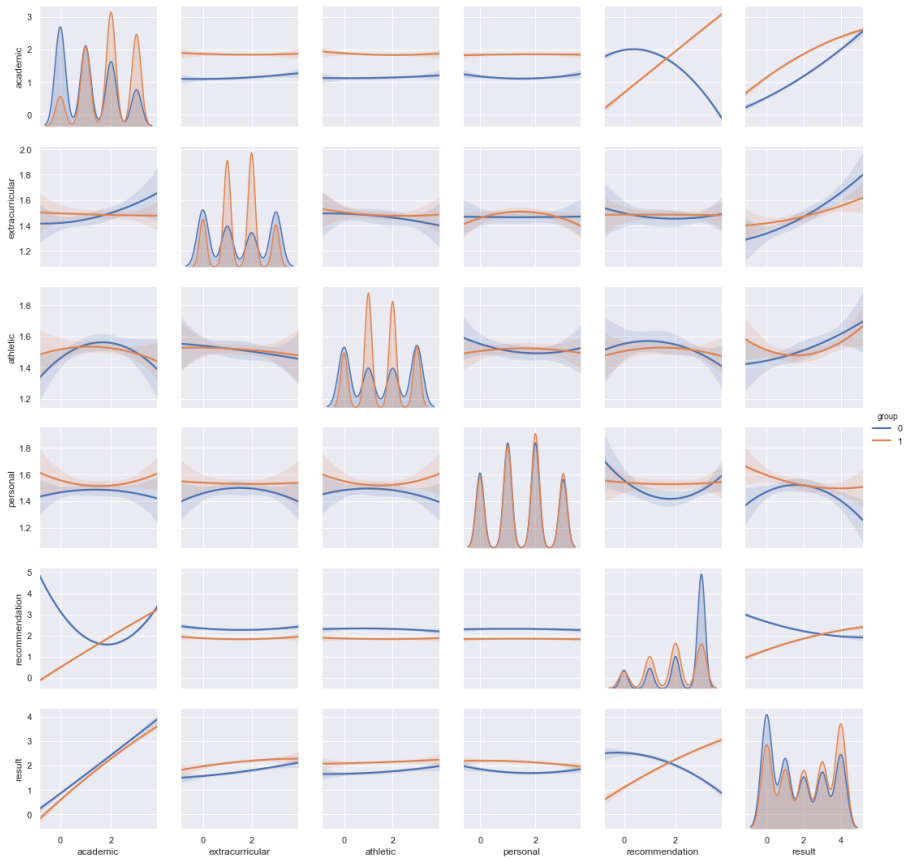


Figure 3.5: Pairplot of a set of 1000 simulated data points grouped by the sensitive variable. Distributions of the variable are on the diagonal while pairwise correlations between the variables are on the off-diagonal. Compare to the data the model was trained on Figure 3.4

that perfectly captures the real world even if we here might be able to create just that since we know the data generating process. Thus it seems far more interesting to explore the performance of this algorithm in the face of imperfect modelling.

CHAPTER 4

Policies

Definition 4.0.1. For a belief θ in the set of possible beliefs Θ, an input x in the set of possible inputs X and an action a in the set of possible actions A, we define a policy as a function $\pi : X, \Theta \to \mathcal{P}(A)$ mapping a belief and an input to a probability distribution over the possible actions.

$\pi(a|x, \theta)$ denotes the probability of taking the action a under the belief θ when observing the input x.

In our case we will use a matrix of weights to to transform an input vector to an output vector, essentially a one-layer neural net, as this mapping. This means that when optimizing our policy we need to find the optimal values for the parameters of our matrix.

We distinguish between two different forms of policies, adaptive and static where they differ in that the adaptive will assign different action probabilities to the same individuals depending on our belief about the world. Thus

$$\pi_{\text{static}}(a|\mathbf{x}', \theta') = \pi_{\text{static}}(a|\mathbf{x}', \theta'')$$

$$\pi_{\text{adaptive}}(a|\mathbf{x}', \theta') \neq \pi_{\text{adaptive}}(a|\mathbf{x}', \theta'')$$

4.1 Transforming the input

Our input $\mathbf{x}$ is a vector of discrete values. In order to enable our policies to give different action probabilities for each possible value of $\mathbf{x}$, even in the case when the dimensionality of $\mathbf{x}$ is 1 and the cardinality is low, we transform each element of $\mathbf{x}$ into a vector representation of the value, Ω through the transformation $\omega : X \to \Omega$. In its simplest form, in the case of 1-dimensional $\mathbf{x}$, this is just a transformation into a one-hot vector

$$\Omega_i = \begin{cases} 1 & i = x \\ 0 & i \neq x \end{cases} \qquad \forall i \in X \tag{4.1}$$

but each policy will use different ω and for the adaptive policies this is typically where θ influences the policy.

4.2 Transforming the output

In order to transform the output from the policy into a probability distribution over actions the output is passed through the Softmax transformation such that

$$\pi(a|x,\theta) = \text{Softmax}_{\text{row}}(\omega(\mathbf{x},\theta)W)$$

where W is the matrix of trainable weights and the Softmax function over each row ensures that the rows, and thus all the action probabilities for each individual, sum to 1.

$$\text{Softmax}(W_{ij}) = \frac{e^{W_{ij}}}{\sum e^{W_{i\cdot}}} \tag{4.2}$$

4.3 Static policy

The primary characteristic of a static policy is that it doesn't use θ in transforming $\mathbf{x}$ into Ω. In it's simplest form it is just Equation (4.1). If we wish to make sure we have some similarity in treatment between adjacent values of $\mathbf{x}_i$ we use

$$\Omega_i = \phi(i - x) \qquad \forall i \in X \tag{4.3}$$

In our case this would give us, for an example input, the following transformation from individual features to policy input, where $\phi(k)$ is the probability density function of the standard normal distribution k standard deviations from the mean:

$$\text{Input: } \begin{bmatrix} 0 \\ 2 \\ 4 \\ 1 \end{bmatrix} \rightarrow X = \begin{bmatrix} \phi(0) & \phi(1) & \phi(2) & \phi(3) & \phi(4) \\ \phi(-2) & \phi(-1) & \phi(0) & \phi(1) & \phi(2) \\ \phi(-4) & \phi(-3) & \phi(-2) & \phi(-1) & \phi(0) \\ \phi(-1) & \phi(0) & \phi(1) & \phi(2) & \phi(3) \end{bmatrix}$$
$$\approx \begin{bmatrix} 0.399 & 0.242 & 0.054 & 0.004 & 0.000 \\ 0.054 & 0.242 & 0.399 & 0.242 & 0.054 \\ 0.000 & 0.004 & 0.054 & 0.242 & 0.399 \\ 0.242 & 0.399 & 0.242 & 0.054 & 0.004 \end{bmatrix}$$

In both these cases W will be $W \in \mathbb{R}^{|X||d|\times|A|}$.

4.4 Adaptive policy

An adaptive policy differs from a static policy in that it uses θ in ω. This allows us to let new information in our simulations of the future lead to different policies and thus hopefully to more information.

One way of doing this is letting $W \in \mathbb{R}^{|Y|\times|A|}$ and

$$\Omega_i = P(y = i|x,\theta) \qquad \forall i \in Y \tag{4.4}$$

CHAPTER 5

Optimization

In order to create the policy that takes the best possible actions we will need to first define what constitutes a good action, and then to create the policy that takes these actions. In Chapter 2 we discussed how to measure the quality of our decisions. Since our policies are neural nets with a set of parameters, our goal is to find the parameter values that results in the best possible decisions according to the defined metrics of fair utility. While we could, in theory, use most optimization algorithms, the most commonly used with neural networks to minimize loss is Stochastic Gradient Descent. In our case we want to maximize fair utility, which means we will use an Ascent instead of a Descent algorithm, or we will invert our metric before runing the algorithm.

We can generalize the optimization process by splitting it into four modules:

1. Main optimization loop
2. Expected Fair Utility function
3. Optimizer stepping function
4. Convergence check

5.1 Main optimization loop

The main optimization loop is the part that controls the other three parts using Algorithm 5. For each iteration it attempts to improve the policy a little bit until there is no further improvement possible. This is done by computing expected fair utility of the policy using one of the algorithms discussed below, then numerically estimating its gradient with respect to the policy parameters before updating the parameters in the direction of higher fair utility.

Algorithm 5: Generalized Gradient Ascent optimizer

begin
  for $i \in 1 : max\ iterations$ **do**
    Set $U_F = E[U_F|\pi, \Theta]$
    For $W \in \mathbb{R}^{m \times n}$ compute $\frac{\partial U_F}{\partial W_{ij}} \quad \forall i \in [1, m], j \in [1 : n]$
    Set $W = \text{step}(W, \frac{\partial U_F}{\partial W})$
    if *convergence* **then**
      break

5.2 Expected Utility functions

We will be using several different kinds of expected utility functions. The two we explore, which we hypothesise will give the best results over time in addition to several established expected utility functions used as baselines for comparison.

Empirical expected utility (aka. Non-Parametric marginal)

Algorithm 6 is the algorithm typically used in most applications of Stochastic Gradient Descent optimizers. It substitutes the difficult question of "what is the expected fair utility of this policy on future data" with the simpler "what is the expected fair utility of this policy on the data where we know the input and the outcome". This makes the assumption that the data we will get in the future will resemble the data we have seen in the past, or, in the case where we have a lot of data (here defined to be more than 1000 data points), that the data we will get in the future will resemble a randomly sampled subset of the data we have seen.

Because this algorithm uses only the empirical fairness and utility on the data seen previously it will, given enough data, give an unbiased estimate of the expected fair utility of the policy given that the data generating process does not change over time. When used as the target function in our optimization it will thus ensure that our policy maximizes the expected fair utility. Its weakness is that the variance of the estimate is very high when we have little data.

Modelled marginal expected utility (aka. Parametric Marginal)

Algorithm 7 is using our models from Chapter 3, but the expected value of the model parameters instead of a sampled value from their distributions.

In contrast to Algorithm 6, Algorithm 7 uses a parametric model of the world to estimate the expected fair utility of a policy. This results in an estimator with far lower variance in low-data settings, but it will be biased in so far as the parametric model is incapable of capturing the intricacies and interactions of the true data generating process. As a result we are optimizing our policy with regard to a different metric than the one we will use to evaluate it in posterity, namely the empirical fair utility.

Algorithm 6: Empirical Fair Utility

begin
- Set $\mathcal{D}' =$ all historical seen data with outcome
- **if** $|\mathcal{D}'| > 1000$ **then**
  - Set $\mathcal{D}_i \sim \mathcal{D}' \qquad \forall i \in [1, 1000]$
- **else**
  - Set $\mathcal{D} = \mathcal{D}'$
- Set $E[U|\mathcal{D}, \pi] = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \sum_A \pi(a|d_{\mathbf{x}}) \cdot U(a, d_y)$
- Estimate $p(a, z|y), p(a|y), p(z|y), p(y, z|a), p(y|a), p(z|a)$ empirically from $\mathcal{D}$ and π
- Set $E[B|\mathcal{D}, \pi] = \sum_{A,Z,Y} |p(a, z|y) - p(a|y)p(z|y)|^P$
- Set $E[C|\mathcal{D}, \pi] = \sum_{A,Z,Y} |p(y, z|a) - p(y|a)p(z|a)|^P$
- Set $E[\bar{F}|\mathcal{D}, \pi] = \gamma E[B|\mathcal{D}, \pi] + (1 - \gamma)E[C|\mathcal{D}, \pi]$
- Set $E[F|\mathcal{D}, \pi] = 1 - E[\bar{F}|\mathcal{D}, \pi]$
- **return** $(1 - \lambda)E[U|\mathcal{D}, \pi] + \lambda E[F|\mathcal{D}, \pi]$

Algorithm 7: Marginal Utility

begin
- Set $\theta' = E[\theta|\Theta]$
- Get $E[U|\pi, \theta')]$ from Equation (2.4)
- Get $E[\bar{F}|\pi, \theta']$ from Equation (2.9)
- Set $E[F|\mathcal{D}, \pi] = 1 - E[\bar{F}|\mathcal{D}, \pi]$
- **return** $(1 - \lambda)E[U|\pi, \theta')] + \lambda E[F|\pi, \theta')]$

Modelled sampled expected utility (aka. Parametric Shallow)

Algorithm 8 is using our models from Chapter 3, sampling parameter values from their distributions. This is largely identical to Algorithm 7, but instead of using the expected value of every parameter in the model it samples randomly parameters from their distributions in order to attempt to calculate the expected fair utility taking into account the uncertainty of our model.

Modelled sampled recursive expected utility (aka. Parametric Deep)

Algorithm 6 tries to optimize the policy for the decision to be made at present, while the Algorithm 7 and Algorithm 8 optimizes the policy in expectation given our model. What new information can we hope to get that makes the optimal policy different that isn't already captured in our model?

One way, albeit computationally expensive, of trying to take into account the possible information we might gather in the future and account for that when we optimize our current policy is to apply a recursive scheme where we sample from our model a set of probabilities, generate data as if these probabilities represent the truth, apply the policy to the new data, update the model with

Algorithm 8: Shallow Utility

begin
- Set $U = \bar{F} = 0$
- **for** $i \in 1 : samples$ **do**
  - Set $\theta \sim \Theta$
  - Get $E[U|\pi, \theta)]$ from Equation (2.4)
  - Get $E[\bar{F}|\pi, \theta]$ from Equation (2.9)
  - Set $U+ = \frac{1}{samples} E[U|\pi, \theta)]$
  - Set $\bar{F}+ = \frac{1}{samples} E[\bar{F}|\pi, \theta)]$
- Set $F = 1 - \bar{F}$
- **return** $(1 - \lambda)U + \lambda F$

the new seen data, then sample from that model again, and go as deep as we choose. This method is detailed in Algorithm 9 and builds on Backward Induction as described by DeGroot (2004, pp. 277-278).

Algorithm 9: Recursive Utility function

begin
- Set $U = \bar{F} = 0$
- Set $U' = \bar{F}' = 0$
- **for** $i \in 1 : samples$ **do**
  - Set $\theta \sim \Theta$
  - Get $E[U|\pi]$ from Equation (2.4)
  - Get $E[\bar{F}|\pi, \theta]$ from Equation (2.9)
  - Set $U+ = \frac{1}{samples} E[U|\pi]$
  - Set $\bar{F}+ = \frac{1}{samples} E[\bar{F}|\pi, \theta]$
  - **if** $depth < targetdepth$ **then**
    - Set $\mathcal{D}' \sim \theta$
    - Set $\mathbf{a} = \pi(a|\mathcal{D}'_x)$
    - Set $\Theta' = Update(\Theta, \mathcal{D}', \mathbf{a})$
    - **for** $i \in 1 : deep\ samples$ **do**
      - Set $U', \bar{F}'+ = \frac{1}{\text{deep samples}} \cdot \text{Self}(\Theta')$
  - Set $U = \frac{(U+U' * \text{discount})}{1+\text{discount}}$
  - Set $F = 1 - \frac{(\bar{F}+\bar{F}' * \text{discount})}{1+\text{discount}}$
  - **return** $(1 - \lambda)U + \lambda F$

The problem with the approach described above is that when we compute an expected fair utility at this point given the model that we have now, the mean (or weighted mean if we want to discount future fair utility) of the expected fair utility for the next T timesteps should be the same. What might change that is to make the policy adaptive,, i.e., have the policy take different actions for the same input if the belief about the world is different. Even so it is hard to see why this would take into account anything the uncertainty in the model

doesn't already.

5.3 Optimizer stepping functions

There are several options in updating our policy to a slightly better version in light of the results from the last iteration. Their differences are mainly in terms of speed, number of iteratrions required to reach optimum and their ability to navigate non-monotonous utility functions.

The most common and the simplest stepper is simply updating the weights using some learning rate and the gradient of the utility function:

$$W'_{ij} = W_{ij} + \eta \frac{\partial U_F(\pi, \Theta)}{\partial W_{ij}} \qquad \forall i \in 1:m, j \in 1:n \tag{5.1}$$

where η is the learning rate and $W \in \mathbb{R}^{m \times n}$.

Other popular options for stepping functions are various momentum-based functions, Adam as proposed by Kingma and Ba 2014 and lately Ranger as proposed by Tong, Liang, and Bi 2019. The main advantage of these is that they use the history of the gradients and the updates to aid in navigating non-monotonous utility functions.

We have tested the convergence ability and speed of four implementations before arriving at Ranger as being the fastest and most stable. The four tested were the one described in Equation (5.1), one using Nesterov Momentum as discussed in Sutskever et al. 2013, Adam and Ranger.

5.4 Convergence

In order to avoid excessive iterations during policy optimization we check, for every 100 iterations, if there has been any significant change in any of the utility measures. This is done by looking at each of the three measures, utility, fairness and their combination, for, if possible, a history of 50, 100, 200, and 500 iterations. This gives us 9, and after 500 iterations 12, loss histories. We then run a Least Squares regression on these histories and if they are not changing over this period with a certain confidence we mark them as passed. If 10 of the 12, or 9 of the 9 checks pass, we determine the optimizer converged.

Algorithm 10: Convergence checker

```
begin
    if iteration > start and (iteration % every) == 0 then
        for history ∈ 50, 100, 200, 500, 2000 do
            Run OLS regression on history of Utility, Fairness and Fair
             Utility last history iterations
            if 0 is in 95% confidence interval of slope then
                Mark loss function and history length as converged
        if Almost all checks have converged then
            Deem optimizer converged
```

5.5 Automating gradient calculation

The trying out different kinds of adaptive policies will require a more complicated differentiation $\frac{\partial \pi(a|x)}{\partial W_{xa}}$. In order to avoid having to do this by hand for every new policy we need to implement a form of automatic differentiation. By using the pyTorch framework we get access to automatic differentiation while we compute our fair utility in addition to the pyTorch implementations of various stepper functions, including several momentum-based steppers, Adam and Ranger.

CHAPTER 6

Experiments

In order to find out whether or not there is an improvement in decision making by applying the principles discussed so far, we will perform a set of experiments. In order to ensure reliable and reproducible results we generated the necessary datasets and saved them so that we could use the same datasets for each experiment. For each setting there are two datasets, one with 5000 individuals in each of 200 separate sets and one with 2000000 individuals in each of 5 separate sets. All the datasets also have 1000000 individuals generated in a separate evaluation set. The smaller dataset was used for testing the methods on small data over successive timeframes while the large dataset was used for creating baselines for what the algorithm would be able to achieve with unlimited data.

With 11 different λ-settings to test for, with 50 timepoints to optimize the policy for, and wanting to run at least 10 tests for each λ this was going to require computational power. Considering also that calculating the utility for every timepoint involved sampling a set of beliefs, simulating data, updating a Bayesian model, calculating utility and sometimes doing so recursively up to 30 layers deep each iteration of the optimizer was going to take time, even after restricting ourselves to closed-form updates of the model and optimizing every aspect of the calculation, computational efficiency was paramount to even being able to generate results at all.

6.1 Preparation - tuning optimizer and metrics

Before we could perform any experiments we had to know that our optimizer would reach the optimal policies for a given set of training data within our framework in a reasonable amount of time. This means selecting the appropriate optimizer. pyTorch supplies vanilla SGD and Adam along with various Momentum-based optimizers, and there is also the separate Ranger optimizer. It also means tuning the early stopping so that we avoid iterating on a converged policy, and to find the right number for the maximum number of iterations in case the early stopping failed. We had to balance the number of iterations with the learning rate, where a lower learning rate in general ensures better convergence, but requires more iterations and thus more time. Finally we had to tune the stochasticity of the optimizer. For the Parametric Shallow Algorithm 8 the number of samples determines the stochasticity in our expected fair utility

estimates. The more parameter values we sample, the less stochasticity we will have, the more we will measure the expected fair utility from the entire distribution, the more time it will take to calculate this, and the slower our iteration speed will be. Thus finding the stochasticity that gives the best convergence speed and the best convergence result was also important.

Optimizing the optimizer, learning rates, number of iterations and early stopping criterion was done through a grid search of optimizing several datasets of different sizes within the range we were going to be optimizing during our experiments. The grid was three-dimensional with learning rates on one dimension, optimizers on another dimension and number of iterations on a third, one being a dynamic number based on the early stopping criterion. This enabled us to determine which combination of optimizer, learning rate and number of iterations resulted in the best and most stable results, as well as which early stopping criterion with which setting most reliably converged at the same result as the optimum and did so the fastest.

Tuning the stochasticity level was then performed with the best setting from above, but in a new grid search with some variation on learning rates and number of iterations as higher stochasticity might require lower learning rates and more iterations.

6.2 Comparing the results

Just producing results for our method without something to compare them to is not very useful. We have therefore generated four different sets of results for comparison by creating two different comparison sets along two different axes.

The first axis is datasize. We generated comparison sets with both one very large dataset in order to determine what possible fair utility we might be able to get if we had infinite data, and in the same manner as we ran the experiments, one small batch of data for each timepoint.

The second axis is how we calculated the expectation. In the method we are investigating we wish to calculate the expected utility and expected fairness violation using the entire model distribution. We wanted to compare this with two other methods, one in which we only used the mean value of each parameter distribution as the parameter, and one in which we used the classic technique when optimizing neural nets - not modelling the world, but assuming that the new individuals will be like the old individuals and therefore substitute calculating the expected fair utility with calculating the empirical fair utility on the individuals we have seen if the policy under evaluation was applied to them. The latter approach has the advantage that you don't need to build a model, which also means you aren't restricted by the model. In the extended setting for example there are interactions in the data that are not modelled. These interactions should be possible to pick up with the empirical fair utility calculation but not with the one that is using the model. It will, however, be more vulnerable to random variation in the data when we have few datapoints. Thus we would expect this empirical approach to perform worse in the low data setting, but to perform better when it has more data.

6.3 Simulating

Algorithm 11: Simulator

begin
1 **for** $\lambda \in$ *list of* λ*s to test* **do**
2 **for** $s \in 1$: *number of simulations* **do**
3 Initialize policy π
4 Initialize model Θ
5 Update Θ with ${}_{T=0}\mathcal{D}$
6 **for** $t \in 1$: *number of timepoints* **do**
7 Optimize π through Algorithm 5
8 Set ${}_{T=t}\mathcal{D}^i_a \sim \pi(a|{}_{T=t}\mathcal{D}^i_{\mathbf{x}})\forall i \in {}_{T=t}\mathcal{D}$
9 Update Θ with ${}_{T=t}\mathcal{D}$
10 Compute $E[{}_{T=t}U, {}_{T=t}F, {}_{T=t}U_F|\pi, \lambda]$ on a large holdout dataset and store

The results have been generated by running Algorithm 11 for each of the three or four different methods for calculating utility at each iteration. They are:

- *Parametric shallow*: This is the model and the method of calculating utility described in Algorithm 8. It uses an explicit Bayesian model and calculates the expected fair utility over that model.
- *Parametric marginal*: This method is similar to the one above, but uses the algorithm described in Algorithm 7.
- *Non-parametric marginal* using Algorithm 6 to calculate utility.
- *Parametric deep* using the algorithm in Algorithm 9. Due to computational cost this was not done 20 times, but 1-6 times if at all.

In addition to this, we generated two baselines to compare the above results to. They are policies trained on very large seen datasets, with 1 million available datapoints, run for only one timepoint (there is no reason to assume they would show any different results on subsequent timepoints) and used to find the upper attainable limit of utility and fairness in the given setting. Ideally then, the low-data experiments should converge to the threshold set by these baselines as they get more data and improve their decisions at each timepoint. There are two, because

1. one uses the Algorithm 7 algorithm during optimization (there is no point in using Algorithm 8 as the parameter distributions should be close to point distributions given this much data) to provide the baseline for the experiments using the modelling-approach, and
2. the other uses Algorithm 6. As the model is an imperfect representation of the world there is reason to believe that it is possible to obtain even

better results using the empirical utility of a policy when training, making it possible for the policy to take into unmodelled interactions between variables.

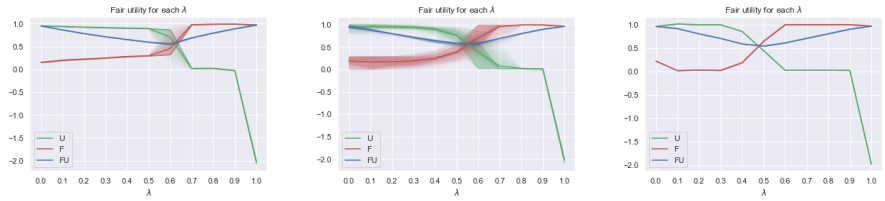
6.4 Low-dimensional with static policy

Policy

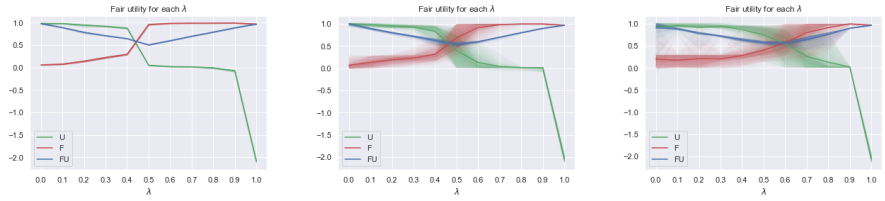
The static policy used in the low-dimensional setting is much as described in Section 4.3. It is a "neural network" with one layer, taking input from each individual as a vector $|X|$ long, that is transformed from a single x-value through Equation (4.3). This vector is then multiplied by a $|X|x|A|$ matrix of learned weights to produce, for each individual, $|A|$ values that are then transformed into action probabilities using the Softmax transformation as described in Equation (4.2).

Results

The results show little difference between any of the utility calculation methods or the baselines, indicating that they all perform quite similarly almost regardless of the amount of data present. The results shown below are only the mean of all the simulations done for each setting. The fair utility calculations used to compare the different approaches below have been calculated as empirical utility and fairness on a large holdout set that is separate for each simulation.

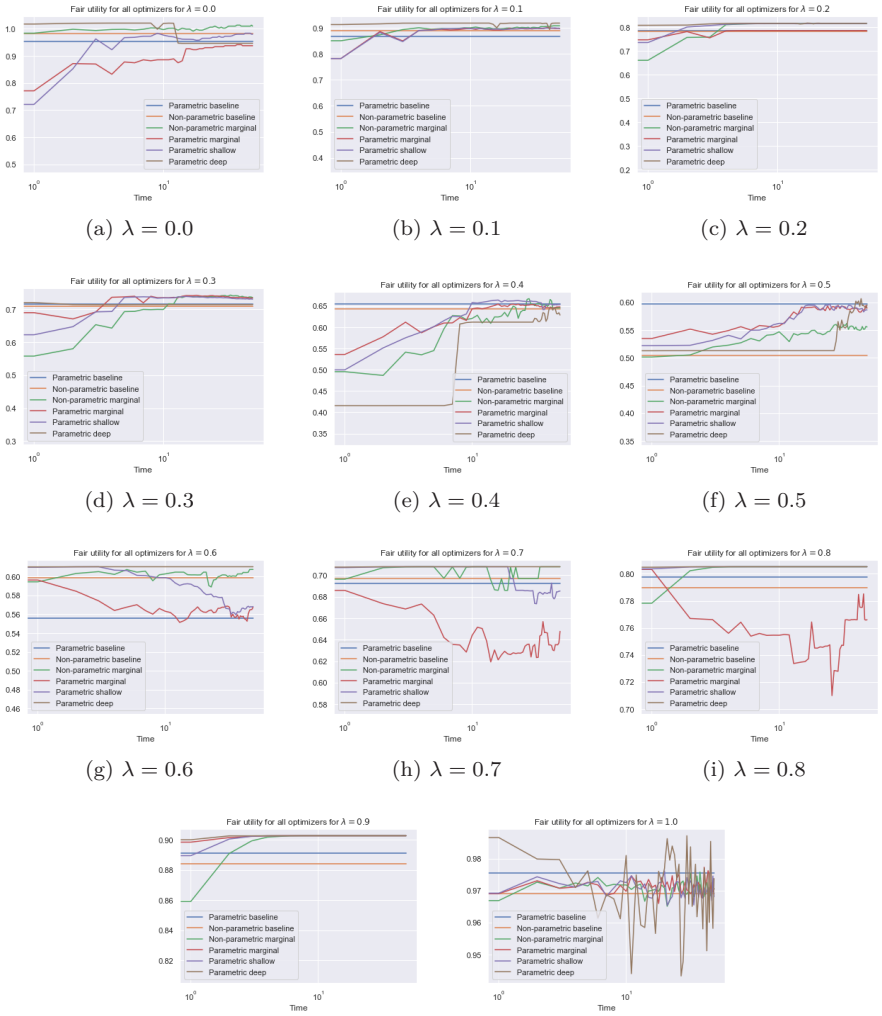


(a) Parametric baseline (b) Parametric shallow (c) Parametric deep



(d) Non-Parametric baseline (e) Non-Parametric (f) Parametric marginal

Figure 6.1: Mean fair utilty with distribution for each of the utility calculations across all λ's



(a) $\lambda = 0.0$ (b) $\lambda = 0.1$ (c) $\lambda = 0.2$

(d) $\lambda = 0.3$ (e) $\lambda = 0.4$ (f) $\lambda = 0.5$

(g) $\lambda = 0.6$ (h) $\lambda = 0.7$ (i) $\lambda = 0.8$

(j) $\lambda = 0.9$ (k) $\lambda = 1.0$

Figure 6.2: Mean fair utility over all simulation for all utility calculations at each timepoint for each λ

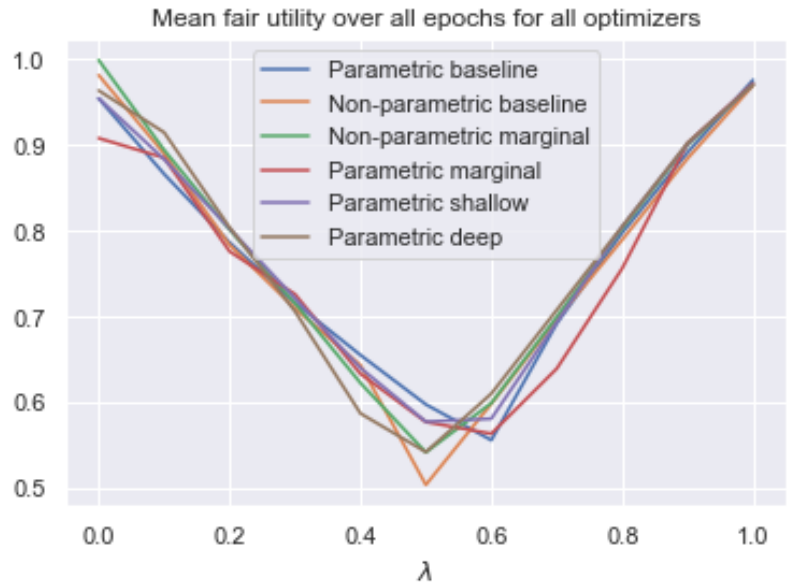


Figure 6.3: Mean fair utility across time and simulations for all optimizers for each value of λ

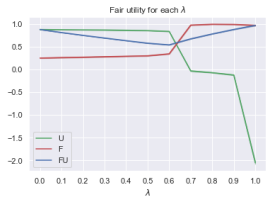
6.5 Low-dimensional with adaptive policy

Policy

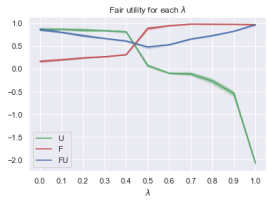
The adaptive policy in the low-dimensional setting is as described in Section 4.4. It is identical to the static policy in that setting with the exception of the transformation from x-value to network input that uses Equation (4.4) instead of Equation (4.3).

Results

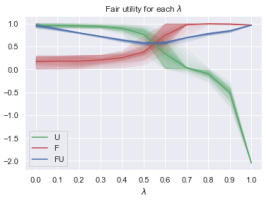
The results show little difference between any of the utility calculation methods or the baselines, indicating that they all perform quite similarly almost regardless of the amount of data present. The results show below are only the mean of all the simulations done for each setting. The fair utility calculations used to compare the different approaches below have been calculated as empirical utility and fairness on a large holdout set that is separate for each simulation.



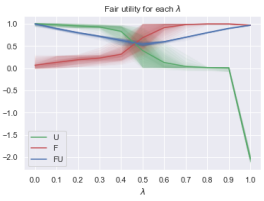
(a) Parametric baseline



(b) Non-Parametric baseline



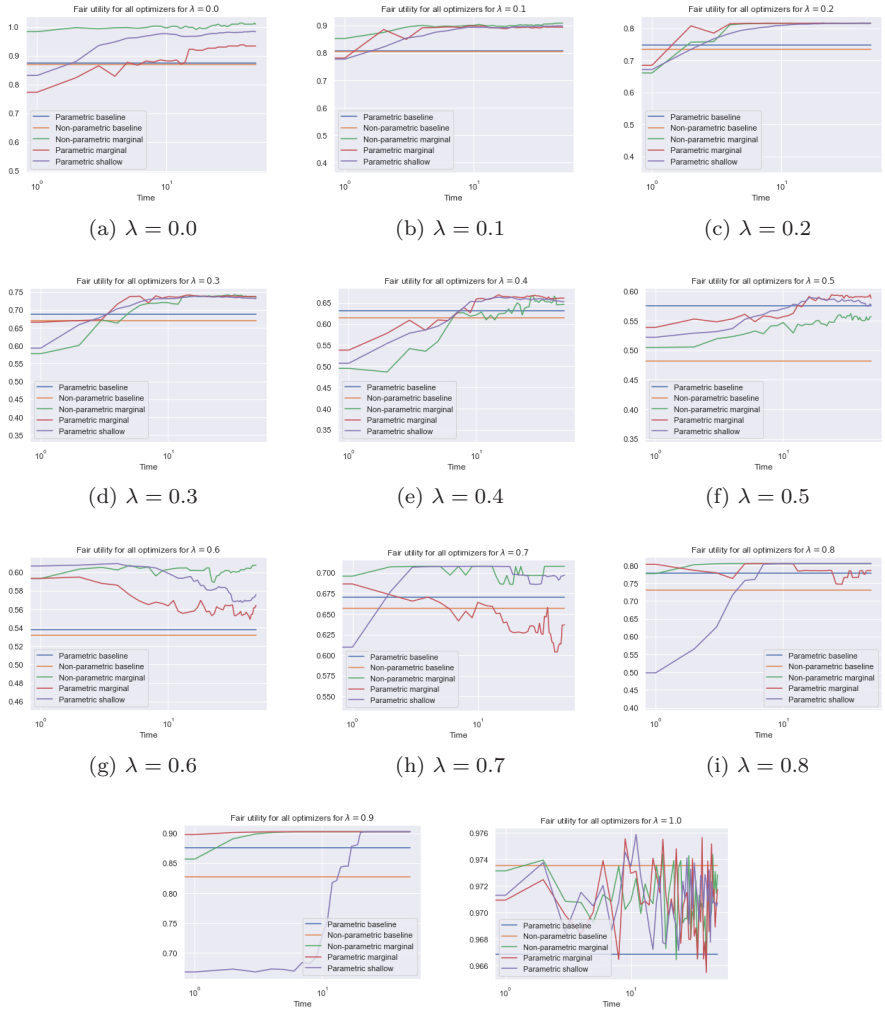
(c) Parametric shallow



(d) Non-Parametric

(e) Parametric marginal

Figure 6.4: Mean fair utilty with distribution for each of the utility calculations across all λ's



(a) $\lambda = 0.0$ (b) $\lambda = 0.1$ (c) $\lambda = 0.2$

(d) $\lambda = 0.3$ (e) $\lambda = 0.4$ (f) $\lambda = 0.5$

(g) $\lambda = 0.6$ (h) $\lambda = 0.7$ (i) $\lambda = 0.8$

(j) $\lambda = 0.9$ (k) $\lambda = 1.0$

Figure 6.5: Mean fair utility over all simulation for all utility calculations at each timepoint for each λ

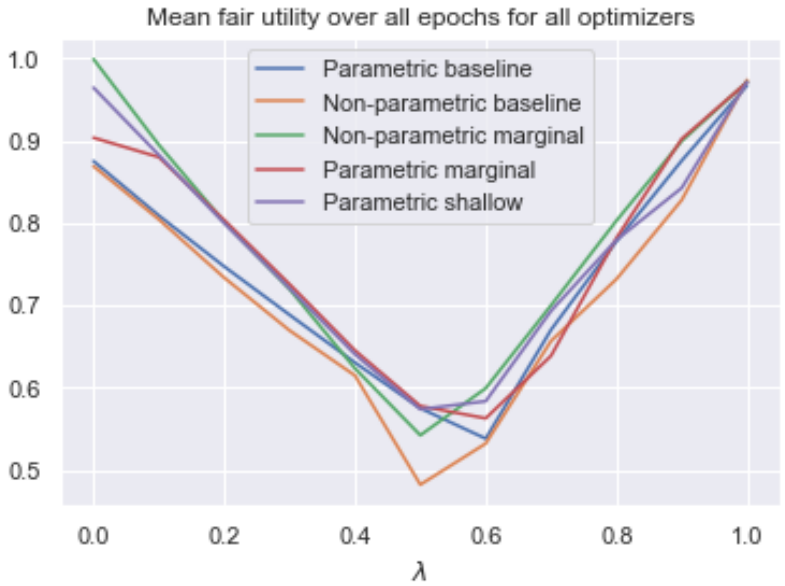


Figure 6.6: Mean fair utility across time and simulations for all optimizers for each value of λ

6.6 High-dimensional with static policy

Policy

In the extended setting the static policy is structured differently. Here the input is no longer one of $|X|_s$ discrete values, but d values from $|X|_e$ possible values. The policy is still a single layer "neural network" without bias, but now the input is not transformed through ω. Thus the input vector $|d|$ long is multiplied with a $|d|x|A|$ matrix to produce the action logits. The big differece between this and the simple static policy is that in this the value of each feature has a linear effect on the action logits, while in the other there is no such linearity.

Thus in this case, we get

$$\pi(a|x,\theta) = \text{Softmax}_{\text{row}}(xW)$$

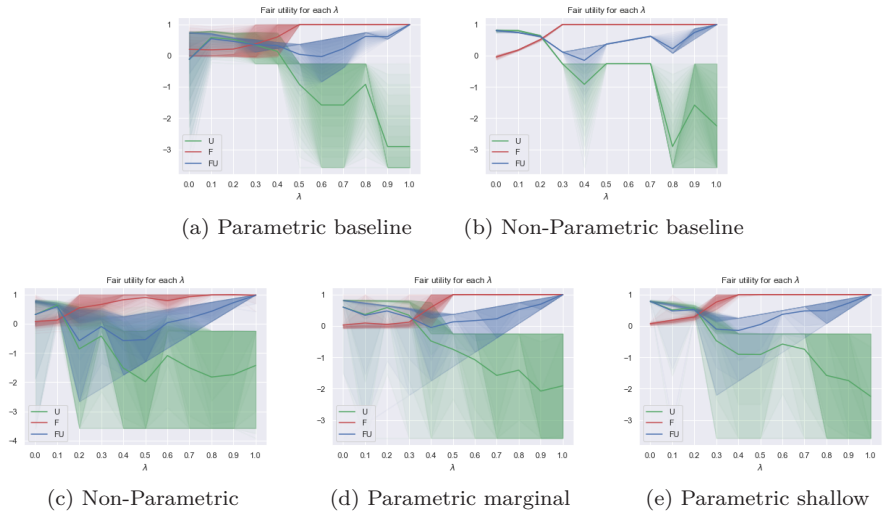
instead of

$$\pi(a|x,\theta) = \text{Softmax}_{\text{row}}(\omega(x)W)$$

Results

The results show huge variance between runs, indicating that the static policy has trouble converging properly.

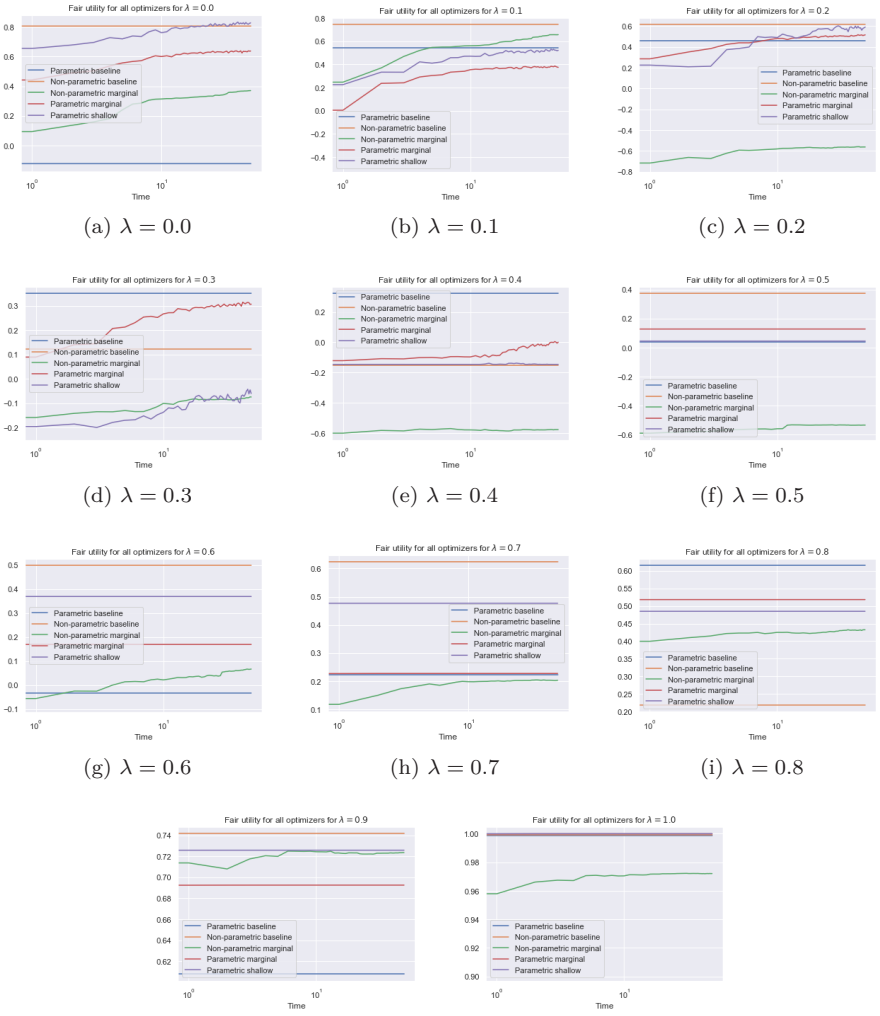
The results shown below comparing the different utility calculations are only the mean of all the simulations done for each setting. The fair utility calculations used to compare the different approaches below have been calculated as empirical utility and fairness on a large holdout set that is separate for each simulation.



(a) Parametric baseline (b) Non-Parametric baseline

(c) Non-Parametric (d) Parametric marginal (e) Parametric shallow

Figure 6.7: Mean fair utilty with distribution for each of the utility calculations across all λ's



(a) $\lambda = 0.0$ (b) $\lambda = 0.1$ (c) $\lambda = 0.2$

(d) $\lambda = 0.3$ (e) $\lambda = 0.4$ (f) $\lambda = 0.5$

(g) $\lambda = 0.6$ (h) $\lambda = 0.7$ (i) $\lambda = 0.8$

(j) $\lambda = 0.9$ (k) $\lambda = 1.0$

Figure 6.8: Mean fair utility over all simulation for all utility calculations at each timepoint for each λ

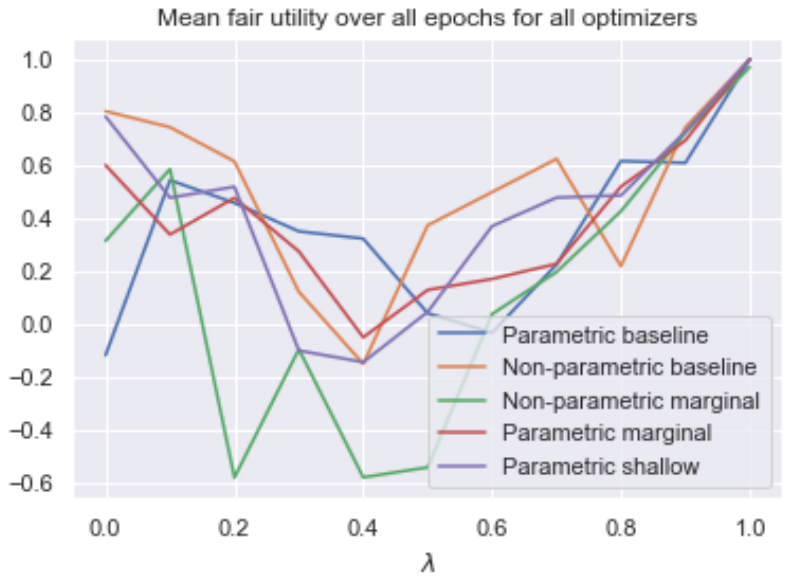


Figure 6.9: Mean fair utility across time and simulations for all optimizers for each value of λ

6.7 High-dimensional with adaptive policy

Policy

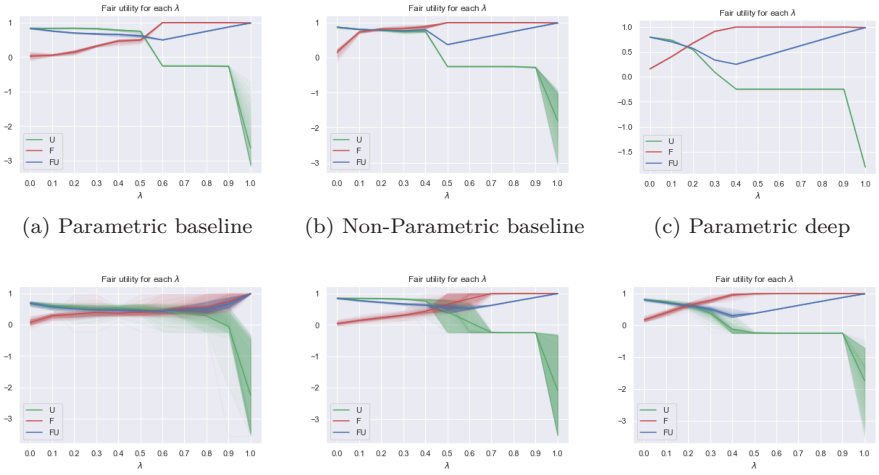
The extended adaptive policy is very different from all the others. This is more of a true neural network, with one hidden layer. As the input transformation it uses Equation (4.1) on each x_i in $\mathbf{x}$ and concatenating the resulting vectors for a $d \cdot |X|$ long vector of 0's and 1's. For the adaptive part it uses an estimate of $P(z|x, \theta$ and $P(y|x, \theta)$ concatenated to the $\mathbf{x}$-input. This gives us a total of $d \cdot |X| + |Z| + |Y|$ input features to the policy. This is reduced to 16 in the hidden layer and $|A|$ in the last.

Results

This is in many ways the most interesting set of results as it shows the results of the presumably best policy, an adaptive one, in the most difficult setting with several dimensions and higher cardinality on X. Here the results show consistently better performance from the parametric marginal utility calculation, on par with the high-data baselines.

This indicates that there is little information to gain in our setting from considering the whole distribution of beliefs. In fact it only adds randomness to our decisions, deviating from optimality without any information to improve future decisions to make it worthwhile.

We also see how the empirical utility calculation has difficulty learning when there is little data relative to the dimensionality of the data. We also see how



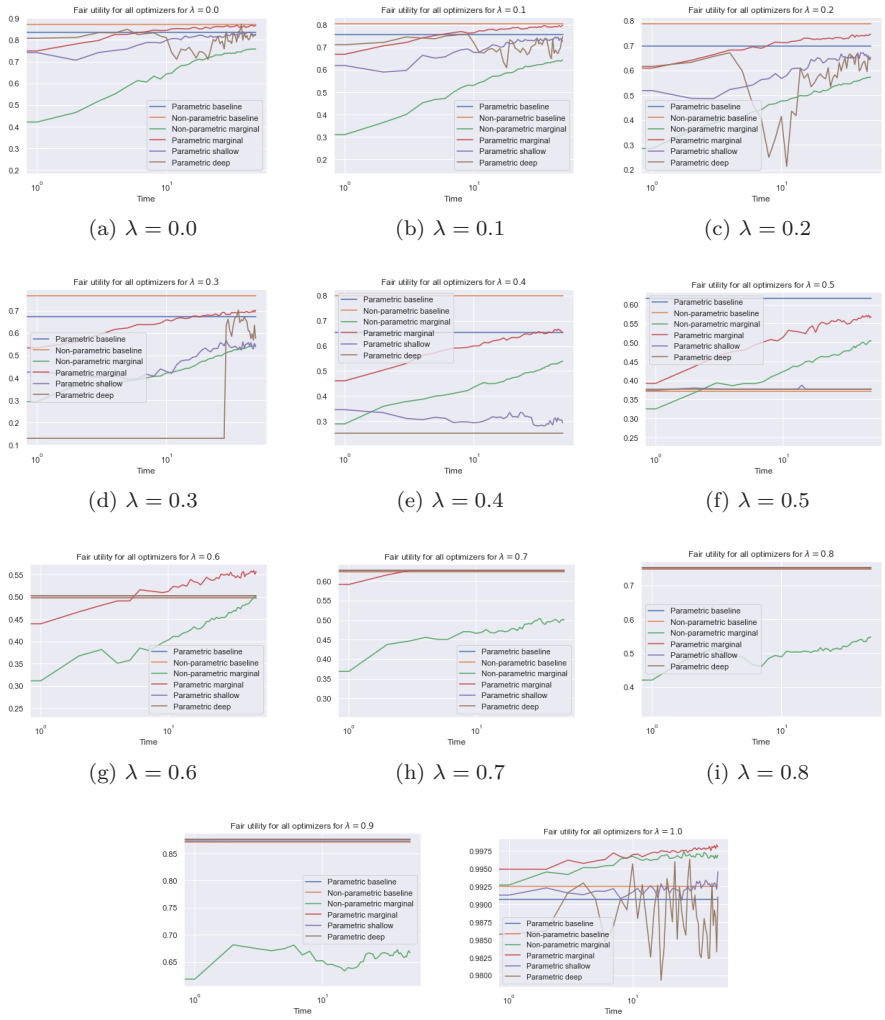
(a) Parametric baseline (b) Non-Parametric baseline (c) Parametric deep

(d) Non-Parametric (e) Parametric marginal (f) Parametric shallow

Figure 6.10: Mean fair utilty with distribution for each of the utility calculations across all λ's

the empirical approach provides the higher baseline, indicating that when it has enough data it will find the patterns and interactions in the data that we could not model explicitly and that allows it to make more precise estimates of the expected fair utility of the policies.

The results shown below comparing the different utility calculations are only the mean of all the simulations done for each setting. The fair utility calculations used to compare the different approaches below have been calculated as empirical utility and fairness on a large holdout set that is separate for each simulation.



(a) $\lambda = 0.0$ (b) $\lambda = 0.1$ (c) $\lambda = 0.2$

(d) $\lambda = 0.3$ (e) $\lambda = 0.4$ (f) $\lambda = 0.5$

(g) $\lambda = 0.6$ (h) $\lambda = 0.7$ (i) $\lambda = 0.8$

(j) $\lambda = 0.9$ (k) $\lambda = 1.0$

Figure 6.11: Mean fair utility over all simulation for all utility calculations at each timepoint for each λ

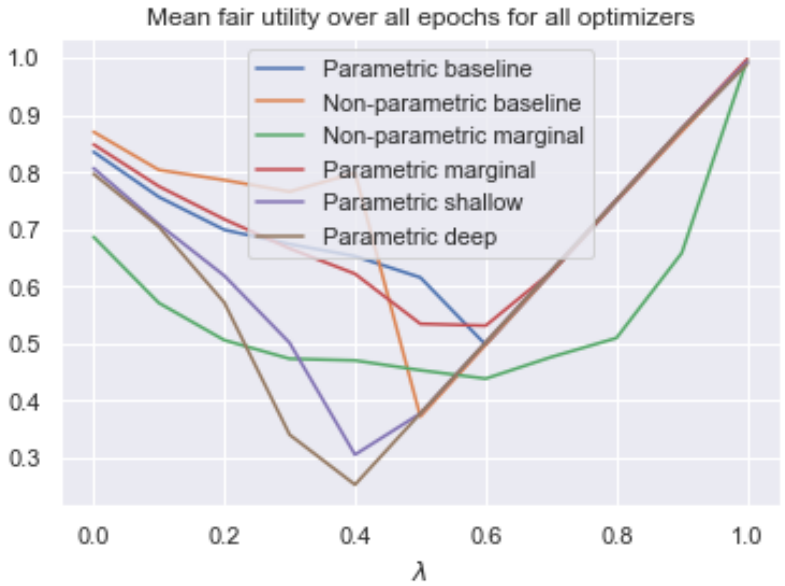


Figure 6.12: Mean fair utility across time and simulations for all optimizers for each value of λ

CHAPTER 7

Conclusion

7.1 Summary of results

In the low-data setting there is much to be gained by using an explicitly modelled approach to estimating our objective functions, whereas in the high-data setting the restrictions imposed by the model makes it impossible to learn unmodelled interactions between variables, causing suboptimal performance. This is analogous to the bias-variance trade-off ubiquitous in machine learning problems and as such not a surprising effect.

Within the modelled approaches that seek to optimize learning we did not see the anticipated effect of improved learning through sub-optimal decisions that we had expected from using the whole model distribution. Rather the information was improved at the same rate as the marginal modelled approach and the decision taken continued to be suboptimal.

7.2 Discussion

Exploration and stochasticity

In hindsight, after seeing the results and writing up everything more carefully, it seems that perhaps the approach of optimizing policy actions for one computed expected fair utility is not going to help us avoid exploting patterns in the first data points we see. We need to define what makes our decision boundary between the two possible actions. It can be one of two, but in either case it is a well-defined boundary.

- We have a limited number l of students we can admit. In this case we will admit if the expected fair utility of admitting the student is positive and higher than $N - l$ other students.
- We have no limit. In this case we only need that the expected fair utility of admitting the student is positive.

In both cases we are just basing our decision on the expected fair utility of the student. In the algorithm for computing the expected fair utility based on the entire distribution of Θ we have some stochasticity, but only in our

estimate of the expected fair utility in each iteration, so with a low learning rate and many iterations it will rather be a way of calculating the expected fair utility by integrating over the full distribution. We will still not introduce any stochasticity into the decision-making process, for example as in Thompson sampling(Thompson 1933), where we could choose a student with lower expected fair utility because we randomly sampled a higher sampled fair utility than another student.

When we use the full distribution we essentially use the entire distribution of $P(y,x)\pi(a|x)U(a,y)$. This means that if $U(a,y)$ was highly non-linear, say

$$U(a,y) = \begin{cases} 0 & a = 0 \\ y & y < 4 \\ 100 & y = 4 \end{cases}$$

we would get a higher consideration of the possibility that a student might get $y = 4$ than when just using the marginal of the Θ distributions.

Another aspect to consider is that the importance of exploration is higher if there are promising students to admit, who will never be chosen because there is someone else to choose that is more promising. We have never in our experimentation explored that kind of setting, and in those settings it will be more important to choose the less promising once in a while. In our setting we have admitted the majority of applicants, meaning that even the least promising mostly get a chance to get admitted, and so there is little information left on the table from low levels of exploring.

In Appendix B we show the results of some experiments using a different kind of policy entirely, that is trained using the marginal and empirical expected fair utility calculations, but when taking actions it is sampling from the models estimated $P(y|x)$-distribution and passing that to the policy.

7.3 Further work

Future work might look at applying the methods described here to more difficult problems to see if that would change the conclusion. It would also be interesting to see alternative optimization techniques used to see if more thorough optimization would lead to better performance of the here tested methods. This would require access to heavier computational power than I had, or further optimization of the code. Examples of more thorough optimization could include learning rate scheduling, lower learning rate, different stochasticity. It would also be interesting to see more experimentation in which settings exploration is more useful and how the stochasticity in the decision-making from out Thompson-inspired policy differs from that of the stochastic policy optimization we use. It would also be interesting to explore the effect a data-generating process that changed over time would have on the usefulness of exploratory techniques.

Appendices

APPENDIX A

The First Appendix

This appendix provides some further details about the implementation and the values used for the various experimentation parameters.

A.1 Utility calculation

The utility function $U(a, y)$ of our students is described in Equation (2.1). It has two terms, $R(y)$ and $\mathcal{C}$ which are detailed below in the form they are used in the experiments as published in Chapter 6.

$$R(y) = \begin{cases} 0 & y = 0 \\ 2 & y = 1 \\ 3 & y = 2 \\ 4 & y = 3 \\ 6 & y = 4 \end{cases}$$

The cost function is more complex than it appears in Equation (2.1). In reality it is a function $\mathcal{C}(\mathbf{a}, \rho)$ giving the cost of a vector of actions on a set of students where ρ is the maximum allowed ratio of applicants to admit. Then

$$\mathcal{C}(\mathbf{a}, \rho) = \sum_{a \in \mathbf{a}} I(a = 1)\mathcal{C}_1 + f(\sum_{a \in \mathbf{a}} I(a = 1), \rho|\mathbf{a}|) \tag{A.1}$$

where $f(\sum_{a \in \mathbf{a}} I(a = 1), \rho|\mathbf{a}|)$ is some monotonous function penalizing admitting more students than there is room for in a way that ensures the gradients of the penalty term is large enough to get the final solution away from the illegal solution space, but not so large as to destabilize the optimizer. The higher the learning rate of the optimizer the narrower the span of stable and useful penalty functions.

In the experiments $\mathcal{C}_1$ was set to 2 and ρ to 1. This reduces $\mathcal{C}(\mathbf{a}, \rho)$ to $\mathcal{C} = 2$ in Equation (2.1).

A.2 More complex networks as policies

One might expect that networks with more layers and more complex architectures might be able to adapt better to the data, but attempts with more information from θ and more complex networks indicate that they rather have trouble converging to the optimal parameters. With only a few value input for each individual there is limited possibilities for learning complex patterns.

A.3 Experiment settings

Below we give a detailed overview of the experiment parameters used

Parameter	Value
optimizer	ranger
learning_rate	0.03(high-dim) 1(low-dim)
max_optim_iterations	30000
gamma	1.0
datasize_per_lookahead ($\|\mathcal{D}'\|$ in Algorithm 9)	100
datasize_per_epoch ($_{T=t}\mathcal{D}$ in Algorithm 11)	20(experiments) 1000000(baselines)
datasize_initial ($_{T=0}\mathcal{D}$ in Algorithm 11)	20(experiments) or 1000000(baselines)
initial_minibatch_size (initial *samples* in Algorithm 8 and Algorithm 9)	4
final_minibatch_size (final *samples* in Algorithm 8 and Algorithm 9, *samples* doubles each time the optimizer converges from initial *samples* until it converges with final *samples*)	16
lookahead_horizon (*targetdepth* in Algorithm 9)	30(for deep) 0(for the rest)
deep_minibatches (*deep samples* in Algorithm 9)	1
future_utility_discount (*discount* in Algorithm 9)	0
future_fairness_discount (*discount* in Algorithm 9)	0
admit_ratio (ρ in Equation (A.1))	1
simulations (number of simulations in Algorithm 11)	20(experiments) 1(deep) 5(baselines)
epochs (number of timepoints in Algorithm 11)	50(experiments) 1(baselines)

Table A.1: Experiment Parameter values

APPENDIX B

The Second Appendix

In light of the considerations explained in Section 7.2, we created a new type of decision-making. Previously when taking actions after optimizing the adaptive policy $\pi(a|\mathbf{x}, \theta)$ we would take actions according to $\pi(a|\mathbf{x}, E[\theta|\Theta])$. Our new decisions are taken according to $\pi(a|\mathbf{x}, \theta \sim \Theta)$, meaning we sample a set of model parameters from Θ, use those to compute the expected reward from admitting each student and assign action probabilities based on that. This is analogous to Thompson sampling(Thompson 1933). It is also quite similar to how we computed action probabilities during optimization when using Algorithm 8, only that we now use it when taking actions.

This framework was tested out in the same fashion as above, but with only the fastest-computed objective functions, Algorithm 7 and Algorithm 6. It was only tested in the high-dimensional setting using the adaptive policy.

B.1 Experiment details

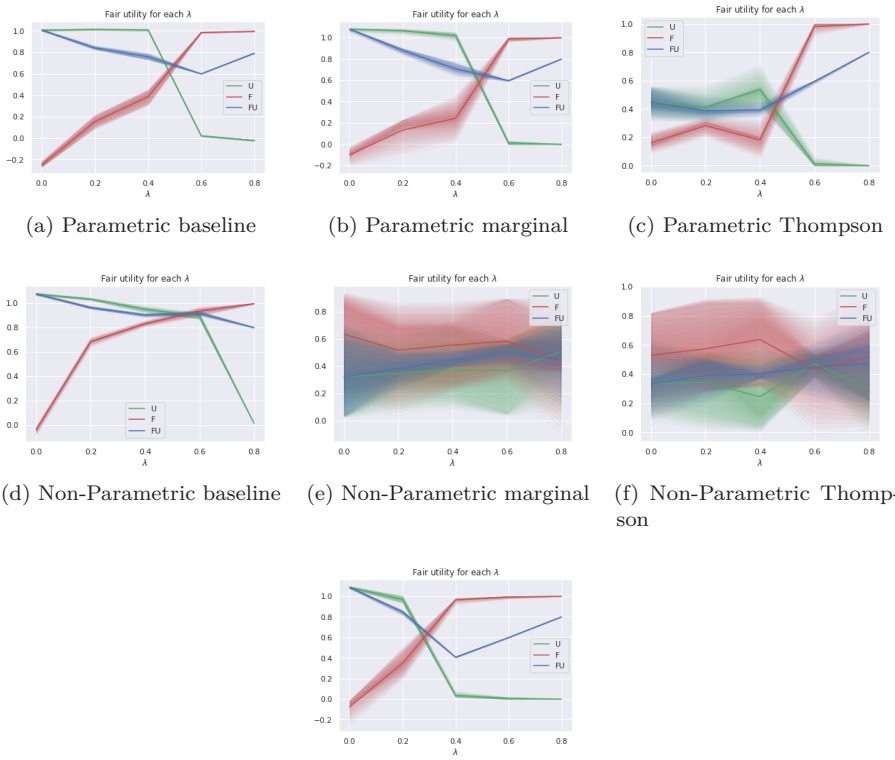
In order to make the problem more difficult along the lines discussed in Section 7.2 we changed a few things. The primary difference is that we changed the reward function to

$$\tilde{R}(y) = \begin{cases} -1 & y = 0 \\ 2 & y = 1 \\ 3 & y = 2 \\ 4 & y = 3 \\ 10 & y = 4 \end{cases}$$

in order to give a large reward for being able to find the best performers. Combined with setting $\mathcal{C} = 5$ we encourage low admittance rates with only the highest y giving a positive utility. We also reduced the number of simulations in order to reduce the computational load.

B.2 Results

We see no advantage to sampling beliefs from their distribution to stochasticize the decision-making process. We have no more learning, and only suboptimal decisions.

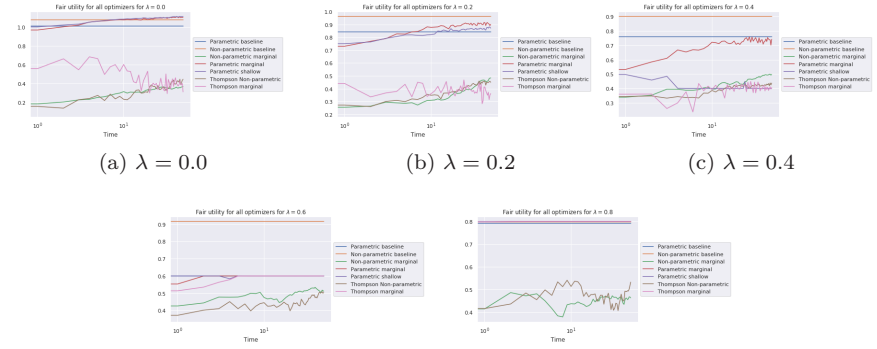


(a) Parametric baseline (b) Parametric marginal (c) Parametric Thompson

(d) Non-Parametric baseline (e) Non-Parametric marginal (f) Non-Parametric Thompson

(g) Parametric shallow

Figure B.1: Mean fair utilty with distribution for each of the utility calculations across all λ's



(a) $\lambda = 0.0$ (b) $\lambda = 0.2$ (c) $\lambda = 0.4$

(d) $\lambda = 0.6$ (e) $\lambda = 0.8$

Figure B.2: Mean fair utility over all simulation for all utility calculations at each timepoint for each λ

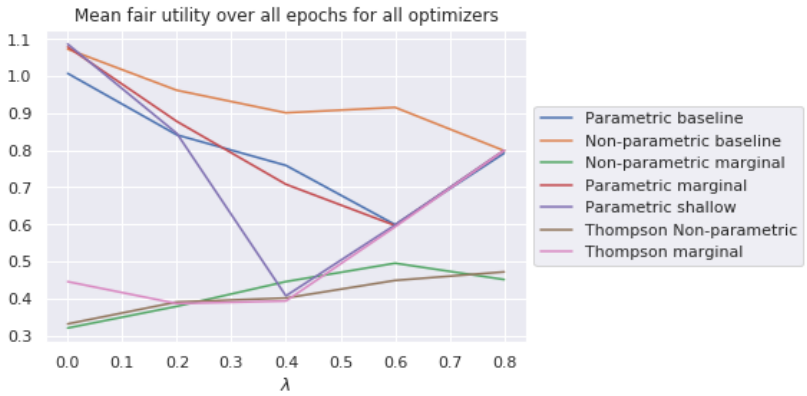


Figure B.3: Mean fair utility across time and simulations for all optimizers for each value of λ

Bibliography

Chouldechova, A. and Roth, A. (2018). "The Frontiers of Fairness in Machine Learning". In: *CoRR* abs/1810.08810. arXiv: 1810.08810. URL: http://arxiv.org/abs/1810.08810.

DeGroot, M. H. (2004). *Optimal Statistical Decisions*. 2nd ed. Wiley-Interscience. John Wiley & Sons, Inc.

Dimitrakakis, C., Liu, Y., Parkes, D., and Radanovic, G. (2017). *Bayesian fairness*. arXiv: 1706.00119 [cs.LG]. URL: http://arxiv.org/abs/1706.00119.

Dwork, C. and Ilvento, C. (2018). "Fairness Under Composition". In: *CoRR* abs/1806.06122. arXiv: 1806.06122. URL: http://arxiv.org/abs/1806.06122.

Fahrmeir, L., Kneib, T., and Lang, S. (2009). *Regression: Modelle, Methoden und Anwendungen*. 2nd ed. Statistik und ihre Anwendungen. Springer-Verlag Berlin Heidelberg, p. 151.

Green, B. and Hu, L. (2018). "The myth in the methodology: Towards a recontextualization of fairness in machine learning". In: *Proceedings of the machine learning: the debates workshop*. URL: https://econcs.seas.harvard.edu/files/econcs/files/green_icml18.pdf.

Grove, W. M., Zald, D. H., Lebow, B. S., Snitz, B. E., and Nelson, C. (2000). "Clinical versus mechanical prediction: A meta-analysis". English (US). In: *Psychological Assessment* 12.1, pp. 19–30. DOI: 10.1037/1040-3590.12.1.19.

Gupta, S. and Kamble, V. (2018). "Temporal Aspects of Individual Fairness". In: *CoRR* abs/1812.04069. arXiv: 1812.04069. URL: http://arxiv.org/abs/1812.04069.

Kearns, M., Roth, A., and Wu, Z. S. (2017). "Meritocratic Fairness for Cross-population Selection". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, pp. 1828–1836. URL: https://www.cis.upenn.edu/~mkearns/papers/FairCDF.pdf.

Kingma, D. P. and Ba, J. (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kleinberg, J. M., Mullainathan, S., and Raghavan, M. (2016). "Inherent Trade-Offs in the Fair Determination of Risk Scores". In: *CoRR* abs/1609.05807. arXiv: 1609.05807. URL: http://arxiv.org/abs/1609.05807.

Kusner, M. J., Loftus, J. R., Russell, C., and Silva, R. (2017). "Counterfactual Fairness". In: arXiv: 1703.06856 [stat.ML].

Liu, L. T., Dean, S., Rolf, E., Simchowitz, M., and Hardt, M. (2018). "Delayed Impact of Fair Machine Learning". In: *CoRR* abs/1803.04383. arXiv: 1803.04383. URL: http://proceedings.mlr.press/v80/liu18c/liu18c.pdf.

Meehl, P. E. (1954). *Clinical versus Statistical prediction*. 1st ed. Vol. 1. 1. University of Minnesota.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*, pp. 1139–1147.

Sutton, R. S. and Barto, A. G. (2020). *Reinforcement Learning: An Introduction*. Second. Adaptive Computation and Machine Learning. The MIT Press, p. 3. URL: http://incompleteideas.net/book/RLbook2020.pdf.

Thompson, W. R. (1933). "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". In: *Biometrika* 25.3/4, pp. 285–294. URL: http://www.jstor.org/stable/2332286.

Tong, Q., Liang, G., and Bi, J. (2019). "Calibrating the Adaptive Learning Rate to Improve Convergence of ADAM". In: arXiv: 1908.00700 [cs.LG].

United States District Court, D. o. M. (2019). *Students for fair admission, Inc v. President and Fellows of Harvard College (Harvard Corp.)* URL: https://www.courtlistener.com/recap/gov.uscourts.mad.165519/gov.uscourts.mad.165519.672.0_2.pdf (visited on 11/06/2019).