CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Bayesian fairness

Master's thesis in Computer Science – algorithms, languages and logic

AMANDA BELFRAGE
DAVID BERG MARKLUND

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Bayesian fairness

AMANDA BELFRAGE
DAVID BERG MARKLUND

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

Bayesian fairness
AMANDA BELFRAGE
DAVID BERG MARKLUND

Supervisor: Christos Dimitrakakis, Department of Computer Science and Engineering
Examiner: Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Bayesian fairness
AMANDA BELFRAGE
DAVID BERG MARKLUD
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis aims to extend the Bayesian fairness algorithm created by Dimitrakakis et al. to be able to handle continuous data. Using bagging to approximate the data we aim to reduce the problem to a computable task that still performs well enough to be an improvement over using the true underlying data. Even though promising results where found for using bagging with discrete data, the continuous version of the algorithm did not work.

Keywords: Fairness, Bayesian fairness, algorithm

Acknowledgements

Thank you Alexander Ågren for your continued support, encouragement, and help during the writing of this thesis. Christos Dimitrakakis we thank you for your guidance, teachings, and neverending patiance. Jonathan Nilsson thank you for your guidance, support, and keeping us on track throughout this process. Squeed we thank you for your support, access to your offices, and all the coffee and tea.

Amanda Belfrage & David Berg Marklund, Gothenburg, January 2019

Contents

List of Figures

1
Introduction

The question of fairness is rather philosophical. To determine what is fair one first has to answer the question on how fairness should formally be defined given a certain setting [1]. This is not a trivial task, but often unfair decisions are made when some kind of private attribute is taken into account when making a decision. To be able to make a fair decision when dealing with people, one idea is to ignore these private attributes, such as gender, race and age, depending on the situation. However, some communities may be inherently disadvantaged, which one also has to take into account. At the same time, a person should not feel that another person in a similar situation got an unfair advantage.

In order to facilitate everyday decision-making processes, such as offering insurance, college admissions and targeted marketing, machine learning-based methods have become increasingly common. As this affects peoples lives, fairness has to be taken into account. There have been made several attempts to solve the problem of including fairness while not affecting the utility, but some of them have a problem with false positives and false negatives. For example Zafar, Valera, Rodriguez, & Gummadi [2] revealed that even though the COMPAS algorithm, an algorithm used to calculate the risk of recidivism within the next two years for a person that has committed a crime, demonstrated similar accuracy regarding the risk of recidivism for white and black people, the direction of errors made for white and black people were different. The errors made on black people were that more people were labeled as high risk for recidivism, while the errors made on white people were that more were labeled as low risk for recidivism. This indicates that the algorithm was biased based on race.

In 2017 Dimitrakakis, Liu, Parkes, & Radanovic presented an idea to handle fairness with an Bayesian viewpoint that focuses on informational aspects on fairness [3]. By taking a Bayesian viewpoint they can take into account model uncertainty, which they claim is crucial for fairness. However, currently it only works for the discrete case. In this thesis we implement the Bayesian fairness algorithm and attempt to extend it to make it work in the the continuous case.

1.1 Aim

The aim for this thesis is to extend the Bayesian Fairness algorithm given by Dimitrakakis et al. [3] to be able to handle the continuous case. As it currently only works in the discrete case the scope for the algorithm is quite limited and this extension would broaden its scope to include more datasets.

1.2 Scope

In this thesis only the possibility to extend the Bayesian Fairness (BF) algorithm to the continuous case will be investigated. It will be compared using the discrete case and a marginal version of the algorithm, which does not take fairness into account. There are a number of other fairness algorithms, but to be able to analyze the possible advantages and disadvantages between them and the BF algorithm, they would also have had to be implemented. Because of time constraints, comparing any other algorithms to the BF Algorithm is out of scope for this thesis and will be left for future work.

The focus with the BF algorithm is to make the extended version work properly. Therefore it is out of scope for this thesis to produce a finished product, i.e. it will not have an interface and there will not be a library ready for release.

1.3 Related work

There have been some attempts to create fair algorithms in the last couple of years. These can be divided into two different categories, algorithms where the outcome is unknown, i.e. the ground truth is unknown, and algorithms where the ground truth is known. When the ground truth is unknown the approaches generally aim to hide or mask the protected attribute, which for example can be race or gender. Zemel et al. maps each individual to a probability distribution in a new representation space, with the aim to get as much information as possible about the individual without revealing its protected attributes [4]. Feldman et al. tries to predict the protected attribute given the results from an algorithm. If they are not successful in predicting the attribute, they consider the algorithm to be fair [5]. If the algorithm is unfair they suggest an approach to repair it by manipulating the protected attribute.

In the other situation, when the ground truth is known, the aim is to prevent the algorithm from making decisions in favor of one group. The risk is that if previous decisions have been unfair it is hard to re-learn the algorithm to make fair decisions. Amazon recently had problems with this. They tried to create an AI that would help with the recruiting. However, it was noticed that the AI was biased against women. The reason for that is that the AI learned by combing through mainly male resumes submitted to Amazon when deciding who to hire. Consequently, the AI

came to the conclusion that men were preferable [6].

One of the most attentive approaches for solving this problem is to match false positive rates (FPR) and false negative rates (FNR) across populations in classification tasks [1]. A post hoc approach on already unfair algorithms, that gives a predictor which is a function of the protected attribute and the original unfair predictor, was presented in 2016 by Hardt et al. The function is probabilistically flipping some of the decisions of the original predictor to match FPR and FNR across populations [7]. However, this approach has it limitations. Woodworth et al. proves that it only works for a strictly convex loss function and an unconstrained hypothesis class, otherwise it might result in a sub-optimal classifier [8]. They therefore conclude that it is necessary to include fairness in the learning process of the algorithms. In the light of that result Zafar et al. proposed an approach that included FPR and FNR matching in the learning process [2], which Bechavod & Ligett had in mind when they developed an algorithm that is based on the idea that unfairness should have a penalty term in the learning process to discourage the algorithm from being unfair [1]. The penalty term they introduced is designed to enforce matching of FPR and FNR.

The BF algorithm is a completely new approach to fairness algorithms. It has indeed enforced fairness in the learning process, but the Bayesian perspective that is adopted by the algorithm is a new perspective and has not, to our knowledge, yet been investigated.

2
Theory

There are a number of different fairness criteria and one can not consider them all at the same time. In this section the different fairness criteria will be defined and the criteria used in this thesis will be presented. A description of the Bayesian fairness (BF) algorithm suggested by Dimitrakakis et al. will also be presented.

2.1 Fairness

To formalize the fairness problem, different kinds of fairness notions have been defined. The two main notions are *disparate treatment* and *disparate impact* [2]. Fairness for disparate treatment means that two subjects differing only on private attributes but having the same public attributes should receive the same judgment, e.g. a woman and a man with the same qualifications should have the same chance to get a job offer. Fairness for disparate impact means that for two groups, where the groups only differ on the private attributes, none of the groups should receive more beneficial or detrimental treatment than the other group. In other words, a group should be treated equally compared to the other group as a whole [2], e.g. it is equally likely for women and men to be misclassified as a good/bad hire.

Kleinberg, Mullainathan, & Raghavan [9] additionally describes three fairness criteria:

- *Calibration within groups*, a set V has a probability P_v that each member of the set belongs to the positive class. For the set V, the expected number of members from group A in V belonging to the positive class should be a P_v fraction of the number of members from A assigned to V.
- *Balance for the negative class* for two groups A, B where Neg_a, Neg_b are all members of the groups belonging to the negative class for each respective group, the average score for each member belonging the negative class should be equal between groups, $avgscore(Neg_a) = avgscore(Neg_b)$. Where the score is the probability that a member should belong to the positive class.
- *Balance for the positive class* is defined similarly as balance for the negative class. The average score assigned to members of group A who belong to the positive class should be the same as the average score of members of group B who belong to the positive class.

The first criterion is connected to disparate treatment, as calibration within groups means that all individuals assigned to V have the same chance of being assigned

to the positive class regardless of which group they belong to, i.e. individuals from groups A, B assigned to set V have the same chance of belonging to the positive class. Each set V, V' with different individuals from groups A, B are divided by what features those individuals present, not what groups they are from. The two balance criteria each represent a form of disparate impact, both the criteria and disparate impact is about how groups are mishandled both negatively and positively.

One can not consider all fairness criteria at the same time as proven by Kleinberg et al. [9]. The two latter criteria are the fairness notions that we focus on in this thesis.

2.2 Bayesian Fairness

The Bayesian Fairness (BF) algorithm suggested by Dimitrakakis et al. is an approach to fairness algorithms under model uncertainty. By adopting a Bayesian viewpoint they incorporate the uncertainty of the model parameters. They claim that taking into account uncertainty has two main effects. Firstly, most notions of fairness are defined with some respects to latent variables, including model parameters. In order to be fair under these notions uncertainty is required. Secondly, many problems are sequential and the data collected is determined by what decision is chosen, taking uncertainty into account ensures that certain small biases will not be magnified in the model [3].

The BF algorithm uses a decision maker (DM) that makes a sequence of decisions based on a policy π and a probability law P, that is assumed to be known, trying to maximize utility u and fairness constraint f:

$$\max_{\pi}(1-\lambda)\mathbb{E}_P^{\pi}u - \lambda\mathbb{E}_P^{\pi}f \tag{2.1}$$

where λ is the trade-off between fairness and utility. The DM has some belief β over a family of distributions $\mathscr{P} \triangleq \{P_\theta \mid \theta \in \Theta\}$ and $\mathscr{P}$ may contain the optimal probability $P_{\theta^*} = P$ for some θ^*. At time t the DM observes some data $x_t \in X$ and depending on the DM's belief at that time, β_t, makes a decision $a_t \in A$ such that $\pi(a_t \mid \beta_t, x_t)$ defines the probability of actions over all possible beliefs and observations. The objective of the DM is to maximize the utility $u : A \times Y \to \mathbb{R}$, where Y is the set of outcomes [3].

The BF algorithm uses the fairness concept of balance as defined by Kleinberg et al. Depending on time t either balance for the negative class or balance for the positive class is used [9]. In the Bayesian setting the amount of information about the model is crucial, less information leads to more uncertainty and in turn a more stochastic decision. By considering the uncertainty of the model the DM can take into account how unfair she would be under all possible models and the probability of those models. This can mitigate some of the problems when working in a sequential setting or

when a small amount of data has been gathered.

Dimitrakakis et al. defines a balanced decision rule $\pi(a \mid x)$, where π is balanced with respect to P_θ if:

$$P_\theta^\pi(a, z \mid y) = P_\theta^\pi(a \mid y) P_\theta^\pi(z \mid y)$$

where P_θ^π is the distribution for P_θ and decision rule θ. Which means that the decision rule is balanced for some distribution P_θ if a, z are independent for all y. This decision rule is based on the balance criteria presented in section 2.1.

Kleinberg et al. also define a calibrated decision rule but it is not used since calibration and balance fairness criteria cannot hold simultaneously [9]. This also holds true for the more general decision rules presented by Dimitrakakis et al. as shown in their paper [3]. Thus we do not consider the calibration rule in this thesis.

Dimitrakakis et al. define a statistical decision problem, see appendix A. We use the same definition in our thesis for the base case of the BF algorithm. In this definition X is discrete, however we want a more general solution where X can be continuous. To achieve that we will use bootstrapping to guess the optimal policy as further discussed in section 2.2.2.

2.2.1 Gradient Calculation

In order to extend the BF algorithm the gradient calculations defined by Dimitrakakis et. al needs to be extended. As the current gradient calculations only works for the discrete case the calculations needs to be extended in order to work in the continuous case. To prevent that the calculations run out of time and space, bootstrapping needs to be integrated with the BF algorithm as well.

Firstly the cost needs to be defined, which is the deviation from balance for a decision rule π under parameter θ. From a family of models $\{P_\theta\}$ with a distribution based on the belief $\beta(\theta)$, the cost, i.e. the deviation from balance, is defined as followed by Dimitrakakis et. al [3]:

$$C(\pi, \theta) \triangleq \sum_{y,z} \| \sum_x \pi(a \mid x) \Delta_\theta(x, y, z) \|_p^q, \tag{2.2}$$

where

$$\Delta_\theta(x, y, z) \triangleq P_\theta(x, z \mid y) - P_\theta(x \mid y) P_\theta(z \mid y)$$

is the balance violation under model θ. Dimitrakakis et. al then concludes that the Bayesian balance of the policy is $f(\pi) = \int_\Theta C(\pi, \theta) d\beta(\theta)$ [3].

The policy for the extended BF algorithm is then defined as the different combinations of the parameters:

$$\pi_\omega(a \mid x) = \omega_{ax}$$

The stochastic gradient descent can then be calculated. First we present the discrete case, defined by Dimitrakakis et. al [3]. They start by defining the inner equation of definition 2.2, as the vector $\mathcal{P}^{\mathcal{A}}$:

$$c_w(y, z) \triangleq \sum_x \pi_w(\cdot \mid x)\Delta(x, y, z),$$

so that they can try to maximize ω for:

$$f_\lambda(w) = u(\beta, \pi_w) - \lambda \sum_{y,z} c_w(y, z)^\top c_w(y, z).$$

Then they calculate the sum and get:

$$\begin{aligned} &\nabla_w (c_w(y, z)^\top c_w(y, z)) \\ &= \nabla_w \sum_a c_w(y, z)_a^2 \\ &= \sum_a 2c_w(y, z)_a \nabla_w c_w(y, z)_a \end{aligned}$$

where:

$$\nabla_w c_w(y, z)_a = \sum_x \nabla_w \pi_w(a \mid x)\Delta(x, y, z)$$

Next they calculate the utility of the equation:

$$\nabla u(\beta, \pi_w) = \int_{\mathcal{X}} d\mathbb{P}_\beta(x) \nabla_w \pi_w(a \mid x) \mathbb{E}_\beta(u \mid x, a).$$

Finally, they combine these two terms into:

$$\begin{aligned} \nabla_w f_\lambda(w) = \int_{\mathcal{X}} &\nabla_w \pi_w(a \mid x)[d\mathbb{P}_\beta(x)\mathbb{E}_\beta(U \mid x, a) \\ &- 2\lambda \sum_{y,z} c_w(y, z)_a \Delta(x, y, z) d\Lambda(x)]. \end{aligned}$$

where Λ is the Lebesgue measure. They end by deriving the gradient for $\nabla_\omega \pi_\omega$ using two different parameterizations. Since these parameterizations are not used for the extension of the BF algorithm they are ignored in this thesis.

For the continuous case of $\mathcal{X}$ the *trivial decision rules for balance* from Dimitrakakis et. al needs to be redefined. Dimitrakakis et. al defines the rule as follows [3]:

$$\sum_x \pi(a \mid x)[P_\theta(x, z \mid y) - P_\theta(x \mid y)P_\theta(z \mid y)] = 0 \tag{2.3}$$

The rule is then redefined for the continuous case:

$$\begin{aligned}
0 &= \int_{\mathcal{X}} \pi(a \mid x) d[P(x, z \mid y) - P(x \mid y)P(x \mid y)] \\
&= \int_{\mathcal{X}} \pi(a \mid x)[P(z \mid y, x) - P(z \mid y)] dP(x \mid y) \\
&= \int_{\mathcal{X}} \pi(a \mid x)[P(z \mid y, x) - P(z \mid y)] \frac{P(y \mid x)}{P(y)} dP(x) \\
&\approx \sum_{x \sim P_\theta(x)} \pi(a \mid x)[P(z \mid y, x) - P(z \mid y)] \frac{P(y \mid x)}{P(y)}
\end{aligned}$$

This allows us to approximate the integral by sampling x, which is needed in the regression case. With this redefinition some of the algorithms needs to be redefined and recalculated. First we redefine $C(\pi, \theta)$ as:

$$C(\pi, \theta) \triangleq \sum_{y,z} \| \sum_{x \sim P_\theta(x)} \pi(a \mid x) \Delta_\theta(x, y, z) \|_p^q, \tag{2.4}$$

with $\Delta_\theta(x, y, z)$:

$$\Delta_\theta(x, y, z) \triangleq [P(z \mid y, x) - P(z \mid y)] \frac{P(y \mid x)}{P(y)} \tag{2.5}$$

The gradients $\nabla_w c_w(y, z)_a$ then needs to be recalculated:

$$\nabla_w c_w(y, z)_a = \sum_{x \sim P_\theta(x)} \nabla_w \pi_w(a \mid x) \Delta(x, y, z) \tag{2.6}$$

For the gradient $\nabla_w f_\lambda(w)$ we are then sampling $x \sim P_\theta(x)$.

Now the policy needs to be recalculated for the continuous case. We have the softmax policy $\pi_w(a \mid x) = \frac{e^{w_a^\top x}}{\sum_{a'} e^{w_{a'}^\top x}}$, for which we define the sum in the denominator as $S = \sum_{a'} e^{w_a^\top x}$. The gradient of the policy is:

$$\nabla_w \pi_w(a \mid x) = \nabla_w \frac{e^{w_a^\top x}}{S}$$

With the different gradients:

$$\nabla_{w_{b,i}} e^{w_a^\top x} = \frac{\partial e^{w_a^\top x}}{\partial w_a^\top x} \frac{\partial w_a^\top x}{\partial w_{b,i}} = e^{w_a^\top x} \mathbb{I}\{b = a\} x_i$$

where $\mathbb{I}\{b = a\}$ is one where $a = b$. The derivative of $w_a^\top x$ then looks like:

$$\nabla w_a^\top x = \frac{\partial w_a^\top x}{\partial w} = \begin{pmatrix} \partial w_a^\top x / \partial w_i \\ \vdots \\ \partial w_a^\top x / \partial w_n \end{pmatrix}_{i=1}^{n} = x$$

where the derivatives are:

$$\frac{\partial w_a^\top x}{\partial w_i} = \frac{\partial \sum_j w_{a,j} x_j}{\partial w_i} = x_i$$

for each i. For the sum S of the softmax policy we have the gradient:

$$\nabla_{w_{b,i}} S = \frac{\partial \sum_{a'} e^{w_{a'}^\top x}}{\partial w_{b,i}} = \frac{\partial \sum_{a'} e^{w_{a'}^\top x}}{\partial w_{a'}^\top x} \frac{\partial w_{a'}^\top x}{\partial w_{b,i}} = e^{w_b^\top x} x_i$$

Putting it all together we get:

$$\begin{aligned}
\nabla_{w_{b,i}} \pi_w(a \mid x) &= \nabla_{w_{b,i}} \frac{e^{w_a^\top x}}{S} \\
&= \frac{\nabla_{w_{b,i}} e^{w_a^\top x} S - e^{w_a^\top x} \nabla_{w_{b,i}} S}{S^2} \\
&= \frac{e^{w_a^\top x} \mathbb{I}\{b = a\} x_i S - e^{w_a^\top x} e^{w_b^\top x} x_i}{S^2} \\
&= \frac{e^{w_a^\top x} \mathbb{I}\{b = a\} x_i}{S} - \frac{e^{w_a^\top x} e^{w_b^\top x} x_i}{S^2} \\
&= x_i \left[\pi_w(a \mid x) \mathbb{I}\{b = a\} - \pi_w(a \mid x) \pi_w(b \mid x) \right]
\end{aligned}$$

we have the general case where we have the partial for w_b:

$$\frac{\partial \pi_w(a \mid x)}{\partial w_b} = x \left[\pi_w(a \mid x) \mathbb{I}\{b = a\} - \pi_w(a \mid x) \pi_w(b \mid x) \right]$$

and the general case for w:

$$\nabla_w \pi_w(a \mid x) = \frac{\partial \pi_w(a \mid x)}{\partial w} = x \left[\pi_w(\cdot \mid x) \mathbb{I}\{b = a\} - \pi_w(\cdot \mid x) \pi_w(b \mid x) \right]$$

where $\mathbb{I}\{b = a\}$ is a one hot encoded vector where $a = b$.

2.2.2 Bootstrapping

Bootstrapping is used to generate classifiers for the DM. The bootstrap algorithm takes as input a training set $\mathcal{D}$ and generates $\mathcal{D}_i$ new training sets by taking samples from $\mathcal{D}$ [10]. To make the re-sampling truly random, the samples are chosen with replacements. That means that every sample that is drawn is put back before another sample is drawn[11]. These new training sets that are created are used to generate a number of classifiers. The resulting set of classifiers are then used by the DM as the family of distributions $\mathscr{P}$.

In our case we use bootstrapping, or simply bagging, as a way to compress the continuous case into smaller more workable cases. By bagging from our continuous $\mathcal{X}$ we are, in a way, taking the continuous case and breaking it down into different distinct discrete cases. Then by checking which of these *bags* that produce the best result we should have a fairly good guess as to what our optimal policy should be.

3
Implementation & setup

To implement the algorithm for the continuous case, bagging first had to be implemented. When that were in place the algorithm could be extended after some recalculations of the model. This section will present the implementation of the extended algorithm in detail. Also, the dataset used for testing as well as the setup for the tests will be described in detail.

3.1 Algorithm Implementation

In order to extend the BF algorithm, bagging needed to be implemented and then the model needed some changes in order to work for the continuous case. In this section a description of the steps taken in order to implement the extended algorithm will be presented in detail.

3.1.1 Bagging

In the bagging process there are two main variables, the amount of bags and the bag size. The amount of bags is the number of bags that are to be created and bag size is the number of data points in the bag, which is usually the same size as the training data. The set of bags will be B containing bags $B_1, \ldots, B_n$ where $\mid B \mid = n$ is the number of bags with the size of each bag $\mid B_n \mid = B_{size}$.

The bag size and bag amount is set in a options file and supplied to the algorithm. The algorithm creates n bags of size B_{size}. Each bag is filled with random data points from the training data as described in section 2.2.2. The true model is created in the same way as before by using the marginal model for the test data.

For the marginal model the full training data is used and calculated in the same way as before. For the Bayesian model however, each bag is used as the basis of its own model with its own model delta. The marginal model is calculated for each of these bags. One of these models is used in each iteration of the stochastic gradient descent (SGD), since each model has a different bag of data the more probable points of data will have a larger impact on each step. This lessens the impact of less uncertain data points of our model. The evaluation is still done on the true model.

3.1.2 Continuous

To extend the algorithm to the continuous case there are some changes that needs to be done to the model. Most of these changes are described in section 2.2.1. A more detailed description of the implementation will be described in this section. First, the input data is normalized to work better with logistic regression. After that the training set and test set is created in the same way as in the discrete case. However, the input data is not compacted like it is in the discrete case, as it is rare that points will coincide within continuous data and compacting it will therefore not work.

In order to create the prior belief some probability matrices for the different probabilities used in the calculations, $P(y), P(y \mid x), P(z \mid y)$, and $P(z \mid y, x)$ needs to be created. For $P(y)$ the frequency of y in the data for each $y \in Y$ is counted. For $P(z \mid y)$ the frequency of z for each y in the data is counted, for each $y \in Y, z \in Z$. To get $P(y \mid x)$ we use logistic regression for y using x. For $P(z \mid y, x)$ we do separate logistic regressions for z using x for each y. Doing this on the training data gives the prior belief.

To calculate the marginal model the frequencies of $P(y)$ and $P(z \mid y)$ were changed into probabilities by checking the percentage of occurrences for y for $P(y)$, and z for each y for $P(z \mid y)$. The probabilities $P(z \mid y)$ and $P(z \mid y, x)$ are already calculated in the prior so those are simply kept in the model. This gives us a marginal model when calculated with the training data. Our true model is created using the test data in the same way.

In order to be able to calculate the gradient, $\Delta(x, y, z)$ needs to be set. As the data could theoretically be infinite, $x \in X$ needs to be sampled from the data. However, in our case the datasets are relatively small and therefore the whole training sets are used. Based on the sampled x's, two more probabilities, $P(y \mid x)$ and $P(z \mid y, x)$, needs to be calculated in order to be able to calculate $\Delta(x, y, z)$. $P(y \mid x)$ is calculated by using the results from the logistic regression of $P(y \mid x)$ and the sampled x's. For $P(z \mid y, x)$ the logistic regressions from the model is used to get the probabilities for each sampled x. Now $\Delta(x, y, x)$ can be calculated as described in section 2.2.1

Lastly, the gradient of $\pi(a \mid x) = S(f(x))$ needs to be calculated, where $S(f(x))$ is the softmax function and $f(x) = Ax$ is the probability vector of each a given x. The gradient for this function is

$$\frac{\partial}{\partial f(x)_c} S(f(x))_y = \frac{\partial}{\partial f(x)_c} \frac{e^{f(x)_y}}{\sum_{c'} e^{f(x)_{c'}}} \frac{\partial f(x)_c}{\partial A_c}$$

where

$$\frac{\partial f(x)_c}{\partial A_c} = x$$

use the following "rule" for

$$\frac{\partial}{\partial f(x)_c} \frac{e^{f(x)_y}}{\sum_{c'} e^{f(x)_{c'}}} : \frac{\partial \frac{g(x)}{h(x)}}{\partial x} = \frac{\partial g(x)}{\partial x} \frac{1}{h(x)} - \frac{g(x)}{h(x)^2} \frac{\partial h(x)}{\partial (x)}$$

This yields:

$$\begin{aligned} &\frac{\frac{\partial}{\partial f(x)_c} e^{f(x)_y}}{\sum_{c'} e^{f(x)_{c'}}} - \frac{e^{f(x)_y} \left(\frac{\partial}{\partial f(x)_c} \sum_{c'} e^{f(x)_{c'}} \right)}{\left(\sum_{c'} e^{f(x)_{c'}} \right)^2} \\ &= \frac{1_{y=c} e^{f(x)_y}}{\sum_{c'} e^{f(x)_{c'}}} - \frac{e^{f(x)_y}}{\sum_{c'} e^{f(x)_{c'}}} \frac{e^{f(x)_c}}{\sum_{c'} e^{f(x)_{c'}}} \\ &= 1_{y=c} S(f(x))_y - S(f(x))_y S(f(x))_c \end{aligned}$$

Which results in:

$$\frac{\partial}{\partial f(x)_c} S(f(x))_y = \left(1_{y=c} S(f(x))_y - S(f(x))_y S(f(x))_c\right) x$$

where c and y are indexes of the elements we want to look at from the output. y is the index of the true value. $1_{y=c}$ is 1 when $y = c$ i.e. when the element of the output corresponds to the true value.

The softmax policy is $\pi_w(a \mid x) = \frac{e^{w_a^\top x}}{\sum_{a'} e^{w_{a'}^\top x}}$. We define the sum in the denominator as $S = \sum_{a'} e^{w_a^\top x}$. The gradient of the policy can be defined as:

$$\nabla_w \pi_w(a \mid x) = \nabla_w \frac{e^{w_a^\top x}}{S}$$

with the sub-gradients:

$$\nabla_{w_{b,i}} e^{w_a^\top x} = \frac{\partial e^{w_a^\top x}}{\partial w_a^\top x} \frac{\partial w_a^\top x}{\partial w_{b,i}} = e^{w_a^\top x} \mathbb{I}\{b = a\} x_i$$

where $\mathbb{I}\{b = a\}$ is one where $a = b$. The derivative of $w_a^\top x$ is:

$$\nabla w_a^\top x = \begin{pmatrix} \partial w_a^\top x / \partial w_i \\ \vdots \\ \partial w_a^\top x / \partial w_n \end{pmatrix}_{i=1}^{n} = x$$

where each derivative is:

$$\frac{\partial w_a^\top x}{\partial w_i} = \frac{\partial \sum_j w_{a,j} x_j}{\partial w_i} = x_i$$

for each i. For the sum S of the softmax we have the gradient:

$$\nabla_{w_{b,i}} S = \frac{\partial \sum_{a'} e^{w_{a'}^\top x}}{\partial w_{b,i}} = e^{w_b^\top x} x_i$$

Which yields:

$$\begin{aligned}
\nabla_{w_{a,i}} \pi_w(a \mid x) &= \nabla \frac{e^{w_a^\top x}}{S} \\
&= \frac{\nabla e^{w_a^\top x} S - e^{w_a^\top x} \nabla S}{S^2} \\
&= \frac{e^{w_a^\top x} \mathbb{I}\{b = a\} x_i S - e^{w_a^\top x} e^{w_b^\top x} x_i}{S^2} \\
&= \frac{e^{w_a^\top x} \mathbb{I}\{b = a\} x_i}{S} - \frac{e^{w_a^\top x} e^{w_b^\top x} x_i}{S^2} \\
&= x_i \left[\pi_w(a \mid x) \mathbb{I}\{b = a\} - \pi_w(a \mid x) \pi_w(b \mid x) \right]
\end{aligned}$$

where the general case when having the partial for w_a is:

$$\frac{\partial \pi_w(a \mid x)}{\partial w_a} = x \left[\pi_w(a \mid x) \mathbb{I}\{b = a\} - \pi_w(a \mid x) \pi_w(b \mid x) \right]$$

and the general case for w is:

$$\nabla_w \pi_w(a \mid x) = \frac{\partial \pi_w(a \mid x)}{\partial w} = x \left[\pi_w(\cdot \mid x) \mathbb{I}\{b = a\} - \pi_w(\cdot \mid x) \pi_w(b \mid x) \right]$$

where $\mathbb{I}\{b = a\}$ is a one hot encoded vector with one where $a = b$.

The gradient calculations for each $x \in X$ and the calculations of the mean of the gradient $\nabla = (1-\lambda)\nabla u - \lambda \nabla f$ to update the policy for the marginal case can now be done. For the continuous case the same calculations are made, except that bagging is used to create different models. For each bag, one model is created using that bag and in the gradient descent we switch between the models taking one step in each models direction. This results in something akin to stochastic gradient descent.

3.2 Dataset

When testing the algorithm, a dataset called the COMPAS dataset were used. The dataset were first filtered, as we did not care about the datapoints with missing values, and the string values were transformed into integer values. Then attributes that were supposed to be protected were chosen and also what attributes were considered were chosen, as not all attributes of the dataset were used.

The COMPAS Dataset[1] contains Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) records from Broward County, Florida (2013-2014). The task for this dataset is to classify whether or not there is risk for recidivism in the next two years. Race and gender was used as protected attributes. As the COMPAS dataset is rather small, we used 70% for training and 30% for testing.

[1] `https://github.com/propublica/compas-analysis`

3.3 Experimental setup

The algorithm was run on the COMPAS dataset multiple times with different input values for each run.Each run used 16 models for the bayesian and bagging approach, as that gave good result with an acceptable speed, and 100 datapoints were used at each iteration. The trade-off between utility and fairness, called lambda, was also set. The value one corresponds to maximum fairness and zero corresponds to maximum utility and the lambdas used were 0, 0.25, 0.5, 0.75 and 1.

4

Results & Discussion

The Bayesian Fairness (BF) algorithm was evaluated on the COMPAS dataset for both the discrete and the continuous case with their respective marginal version as baseline. In this section the results from those runs are presented.

4.1 Bagging in the discrete case

In the discrete case when using two protected attributes, the results shows that bagging performs in between the marginal and BF algorithm when fairness and utility is taken under equal consideration, see Figure 4.1.

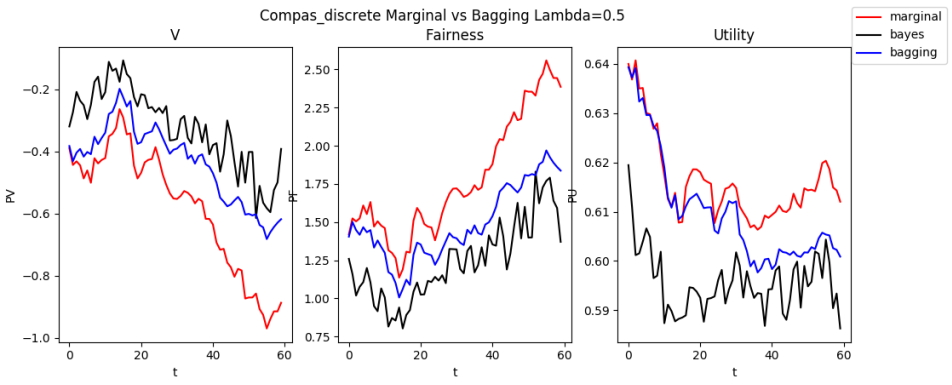

Figure 4.1: Compas data using two protected attributes. Test with discrete models showing the differences of the different approaches. Using 16 models for bayes and bagging. X-axis shows step t where each step is 100 datapoints. The plot shows objective (V), fairness (F) and utility (U). The plot shows test run for lambda 0.5.

When separating fairness (F) and utility (U), the bagging version of the BF algorithm shows no difference compared to the marginal and the bayes version when it comes to utility. Looking at fairness it performs in between the marginal and bayes, see Figure 4.2.

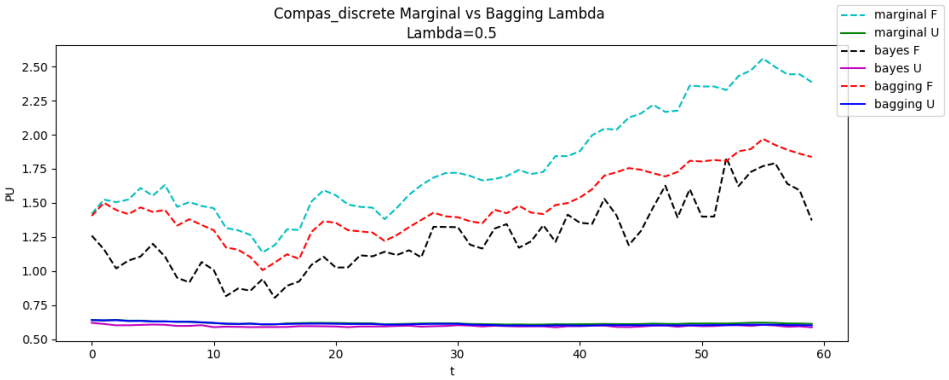

Figure 4.2: Compas data using two protected attributes. Same test settings as 4.1 showing fairness and utility for each approach. X-axis show step t where each step is 100 datapoints.

4.2 Bagging in the continuous case

The results presented in this sections shows how well the BF algorithm performs on the COMPAS dataset when bagging is applied in the continuous case compared to the marginal, with different weight placed on fairness.

As shown in Figure 4.3, the result when fairness is not taken into consideration at all, there are no significantly difference between the marginal and the BF algorithm.

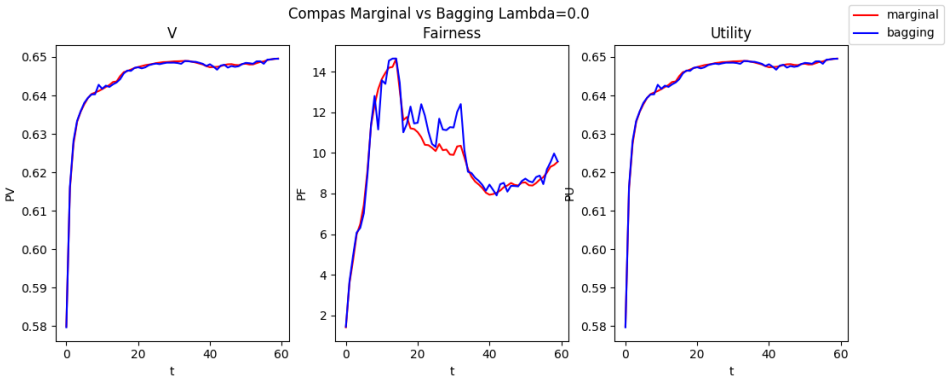

Figure 4.3: Compas data. Test using continuous models. Test run with lamba 0 (Maximize utility), 16 models for bagging. X-axis shows step size of 100 datapoints.

Figure 4.4 shows that the BF algorithm performs slightly better than the marginal when taking some fairness into account, here the objective is 25% fairness and 75% utility. Looking only at the fairness result it shows that the BF algorithm performs slightly better than the marginal, same goes for the utility.

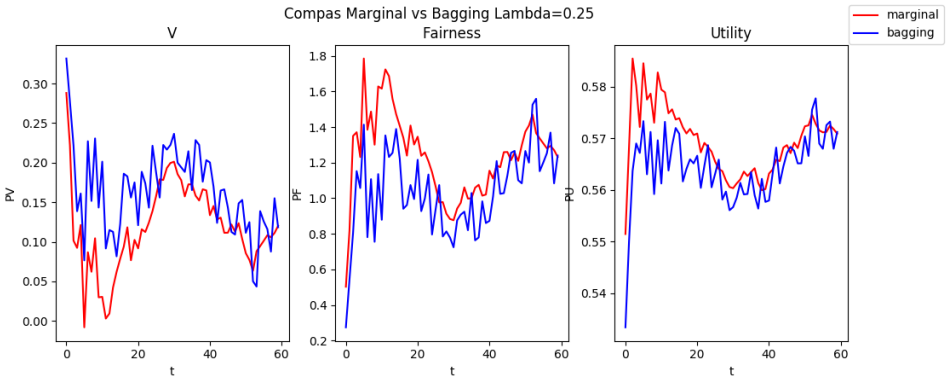

Figure 4.4: Compas data. Test using continuous models. Test run with lamba 0.25, 16 models for bagging. X-axis shows step size of 100 datapoints.

When the objective is set to take fairness and utility under equal consideration, i.e. fairness and utility are 50% each, figure 4.5 shows that there is not much of a difference compared to the previous one, figure 4.4. Looking only at fairness the BF-algorithm does not perform much better than the marginal and the utility keeps being slightly better compared to the marginal.

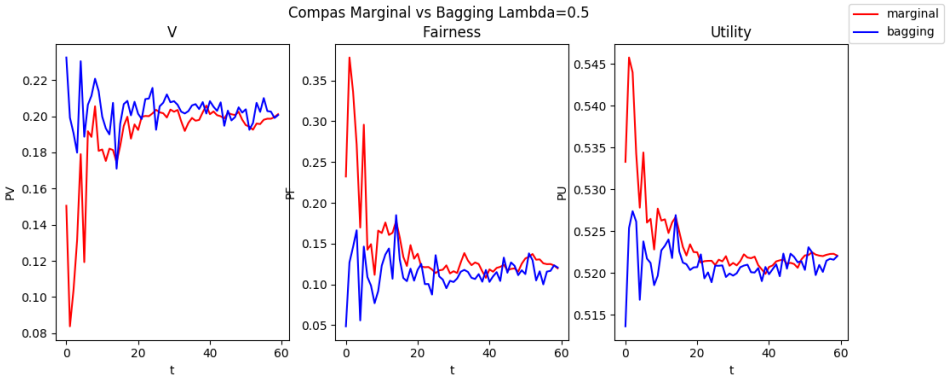

Figure 4.5: Compas data. Test using continuous models. Test run with lamba 0.5, 16 models for bagging. X-axis shows step size of 100 datapoints.

Increasing the fairness to take under consideration to 75% gives the result shown in Figure 4.6. Once again the result is not much different compared to the others presented above. The BF algorithm performs slightly better on both fairness and utility compared to the marginal.

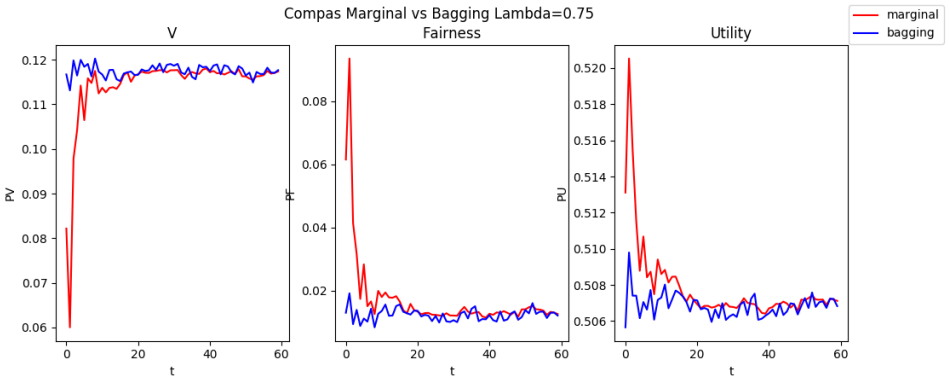

Figure 4.6: Compas data. Test using continuous models. Test run with lamba 0.75, 16 models for bagging. X-axis shows step size of 100 datapoints.

The last objective that were tested was when the BF algoirthm should only take fairness into consideration and no utility, i.e. fairness was set to 100% and utility to 0%. In that case there is no difference to the marginal, as shown in Figure 4.7.

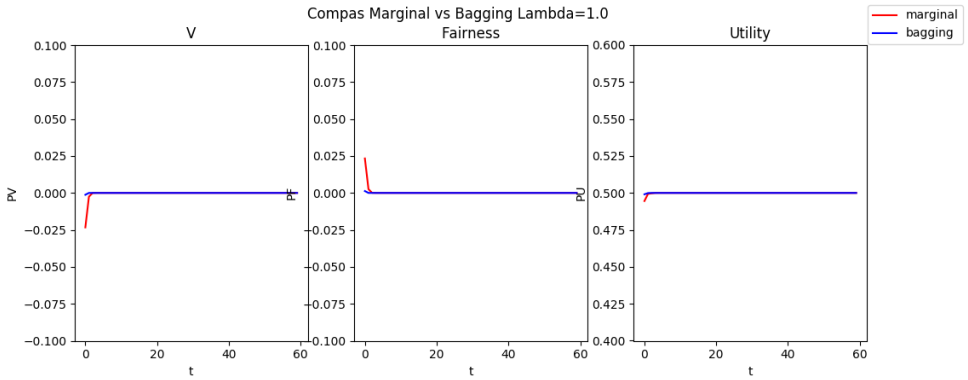

Figure 4.7: Compas data. Test using continuous models. Test run with lamba 1 (maximize fairness), 16 models for bagging. X-axis shows step size of 100 datapoints.

4.3 Discussion

As the BF algorithm originally only worked for the discrete case, the first step was to implement bagging on that. As presented in the results, the BF algorithm extended with bagging performed well in that case. It was not quite as good as the BF algorithm without bagging, but it did perform better than the marginal. That gave hope that bagging could be an acceptable substitute in the continuous case. However, as the results presented, bagging did not perform as expected for the continuous case and does not give a considerable advantage looking at fairness compared to the marginal.

We can see for lambda 0 and 1 that both approaches behave close to equal, which is expected when maximizing for either fairness or utility. This is mirrored in the

graphs with the only caveat that in the lambda 0 case, fairness is slightly different but in such a minuscule way that its not really interesting. For the other cases we can see that bagging outperforms the marginal in all cases in a small way, mostly by just being a bit better overall but then converging toward the same point as the marginal. This should not be the case. As more weight is placed on guaranteeing fairness, we should be able to see bigger difference, and a more fair result, but no such results were found. This means that bagging does not give a noteworthy improvement to the marginal as it does in the discrete case.

5
Conclusion

In this thesis we tried to extended the BF algorithm to work for the continuous case by implementing bagging. The results points to that this is not a valid solution as it did not performed as expected. However, there are other approaches to solve this problem so some future work could be to investigate which solution is best, for example Bayesian logistic regression may be an alternative.

Our finding suggest that the Bayesian Fairness algorithm gets about the same result when bagging is applied compared to the original BF algorithm in the discrete case. However, when applied to the continuous case the BF algorithm does not perform better than the marginal looking at fairness. We have therefore come to the conclusion that bagging is not an alternative when extending the BF algorithm.

As the algorithm would be even more powerful with the extension, since it then can be applied in more cases, our findings are still of importance. As it is now known that bagging is not an alternative, other approaches can be sought out.

Looking at other fair algorithms that have been presented, one of the biggest advantages of the BF algorithm is that it can take more than one protected attribute. This could in some cases be crucial to be able to make a fair decision. However, the algorithm is not yet perfect as it still has a little effect on utility. This could maybe be the result of the settings which could be tweaked even better. The algorithm would also still benefit from an extension that make it work in the continuous case. A future step could be to implement another solution for that. For example Bayesian logistic regression may be a better approach.

Bibliography

[1] Y. Bechavod and K. Ligett, "Penalizing unfairness in binary classification", *arXiv preprint arXiv:1707.00044*, 2017. [Online]. Available: https://arxiv.org/pdf/1707.00044.pdf.

[2] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi, "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment", *CoRR*, 2017. arXiv: 1610.08452. [Online]. Available: https://arxiv.org/abs/1610.08452.

[3] C. Dimitrakakis, Y. Liu, D. C. Parkes, and G. Radanovic, "Subjective fairness: Fairness is in the eye of the beholder", *CoRR*, vol. abs/1706.00119, 2017. arXiv: 1706.00119. [Online]. Available: http://arxiv.org/abs/1706.00119.

[4] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations", in *International Conference on Machine Learning*, 2013, pp. 325–333.

[5] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact", in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 259–268.

[6] I. A. Hamilton. (). Amazon built an ai to hire people, but reportedly had to shut it down because it was discriminating against women, [Online]. Available: https://nordic.businessinsider.com/amazon-built-ai-to-hire-people-discriminated-against-women-2018-10. (accessed: 19.11.2018).

[7] M. Hardt, E. Price, N. Srebro, *et al.*, "Equality of opportunity in supervised learning", in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[8] B. E. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro, "Learning non-discriminatory predictors", *CoRR*, vol. abs/1702.06081, 2017. arXiv: 1702.06081. [Online]. Available: http://arxiv.org/abs/1702.06081.

[9] J. M. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores", *CoRR*, vol. abs/1609.05807, 2016. arXiv: 1609.05807. [Online]. Available: http://arxiv.org/abs/1609.05807.

[10] L. Breiman, "Bagging predictors", *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996, ISSN: 1573-0565. DOI: 10.1007/BF00058655. [Online]. Available: https://doi.org/10.1007/BF00058655.

[11] T. Hesterberg, D. S. Moore, S. Monaghan, A. Clipson, and R. Epstein, "Bootstrap methods and permutation tests", *Introduction to the Practice of Statistics*, vol. 5, pp. 1–70, 2005.

A

Appendix A

The statistical decision problem is defined as follows by Dimitrakakis et. al:
'The DM observes $x \in X$, then takes a decision $a \in A$ and obtains utility $u(y, a)$ depending on a true (latent) outcome $y \in Y$ generated from some distribution $P_\theta(y \mid x)$. The DM has a belief $\beta \in B$ in the form of a probability distribution on parameters $\theta \in \Theta$ on a family $\mathscr{P} \triangleq \{P_\theta(y \mid x) \mid \theta \in \Theta\}$ of distributions. In the Bayesian case, the belief β is a posterior formed through a prior and available data. The DM has a utility function $u : Y \times A \to \mathbb{R}$, with utility depending on the DM's action and the outcome.
For simplicity, we will assume that X, A, and Y, are finite and discrete, whereas Θ will be a subset of $\mathbb{R}_n$. We focus on Bayesian decision rules, i.e. rules whose decisions depend upon a posterior belief β.' [3, p.5]