A non-convex optimization based framework for fairness

A thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Vivek Vardhan Adepu
(17CS10002)

Under the guidance of
Dr. Niloy Ganguly

Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
West Bengal, India
May, 2021

Certificate

*This is to certify that the work contained in this thesis entitled "**A non-convex optimization based framework for fairness**" is a bonafide work of **Vivek Vardhan Adepu (17CS10002)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur under my supervision and that it has not been submitted elsewhere for a degree.*

Dr. Niloy Ganguly
Professor
Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur,
West Bengal

May, 2021
Kharagpur

Abstract

As Artificial Intelligence is becoming more and more ubiquitous, there has also been a rise in the problems caused by them. One of the main issues is lack of fairness in some applications like automatic recruiting system, image recognition, etc. These algorithms tend to directly bias based on some sensitive attributes like sex, race, region, etc or indirectly based on other attributes. Fairness in Machine Learning is a quite a popular topic and there has been a lot of research in the past few years. Despite this, there is no one single theory of algorithmic fairness. In this paper, we present a non-convex optimization framework for ensuring algorithmic fairness, using combinatorial discrepancy. We also discuss already existing works and draw comparisons with them

Contents

Chapter 1

Introduction

Bias is pervasive in the society. We tend to discriminate against some race, religion, caste, etc, in various domains like recruitment, funds provision, recidivism, etc, based on our prejudices and preconceived notions of them. To tackle this, we might consider Machine learning as a feasible solution. But, those do fall prey to discriminatory bias

For example, Safiya Umoja Noble in her book, *Oppression: How Search Engines Reinforce Racism* [1], drew comparisons between search results for *'white girls'* and *'black girls'*. She said that searching for *'black girls'* resulted in sexual content, which is demeaning of them. Another issue, with the google ads, was pointed out by Latanya Sweeney in her paper *Discrimination in Online Ad Delivery* [2]. As per the study, searching for black names generated ads containing word "arrest" more often than searching for white names on some websites. Nikon's face detection algorithm cameras were accused of being racist in 2010 [3]. It detected a Taiwanese-American and her family's eyes as blinked, when they are trying to take a picture, even though they did not blink. In another case, when COMPAS(Correctional Offender Management Profiling for Alternative Sanctions) was analyzed for bias, it was found that it predicted high recidivism rates for black prisoners than white prisoners [4]

In this project, we study and implement a different notion of algorithmic fairness based on discrepancy theory and draw comparisons with the similar implementations

Chapter 2

Background

2.1 Convex optimization

An optimization problem is of the form [5]:

$$\begin{aligned} &minimize \;\; f_0(x) \\ &subject\ to \;\; f_i(x) \leq b_i, \;\; i = 1, ..., m \end{aligned} \tag{2.1}$$

where, $x = (x_1, ..., x_n)$ is the optimization variable, the function $f_0 : R^n \mapsto R$ is the objective function, the functions $f_i : R^n \mapsto R$, i = 1, ..., m, are the constraint functions, and the constants b_i, i = 1, ..., m are the limits, or bounds, for the constraints.

Convex optimization is a special case in which objective function and constraint functions are convex functions i.e. they satisfy the following condition:

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{2.2}$$

There are several methods like interior points, gradient descent, etc. to solve these problems. But, here we are dealing with a non-convex optimization problem which cannot be solved by those.

2.2 Combinatorial discrepancy

Given a set X consisting of n elements and a collection $S = \{S_1, S_2, ..., S_m\}$ of subsets of X, in combinatorial discrepancy, we try to find a coloring $\chi : X \mapsto \{1, 1\}$ such that

discrepancy of the system is minimized.

$$D(X, S) = \min_{\chi \in \{-1,1\}^n} \max_{S_i \in S} D(S_i) \tag{2.3}$$

where,

$$D(S_j) = \left| \sum_{a \in S_j} \chi(a) \right| \tag{2.4}$$

We extend the notion of combinatorial discrepancy [6] to fairness for machine learning algorithms. Given a data set, we first form subsets $S_i^j \in S$, which contain elements of X(full data set) that have a value j of the sensitive attribute i.

For example, let's say i corresponds to sensitive attribute 'gender', j = 1 represents *male* and j = 0 represents *female*, then S_i^1 contains the elements corresponding to *male* and S_i^0 contains the elements corresponding to *female*. Intuitively, this means that we are trying to ensure each candidate has equal opportunity to get each color, job opportunity in case of recruitment system

2.3 Support Vector Machines

"Support vector machines are a set of supervised learning methods used for classification, regression and outliers detection" [7]. Support vector machines are used in wide range of applications like face detection, image classification, Generalized predictive control, Speech recognition, etc

Initially designed as a binary classifier, it is extended to multi-class classification and regression. A support vector machine finds a hyperplane that distinctly classifies the data points. The points falling on the either sides belong to different classes. As there are many possible hyperplanes, it picks the one with the maximum margin, i.e the maximum distance between data points of both the classes.

Support vectors are the points in the data that are furthest from other points in each class. Margin can also be defined as the distance between the support vectors(M in Fig. 2.1). SVM tries to maximize this.

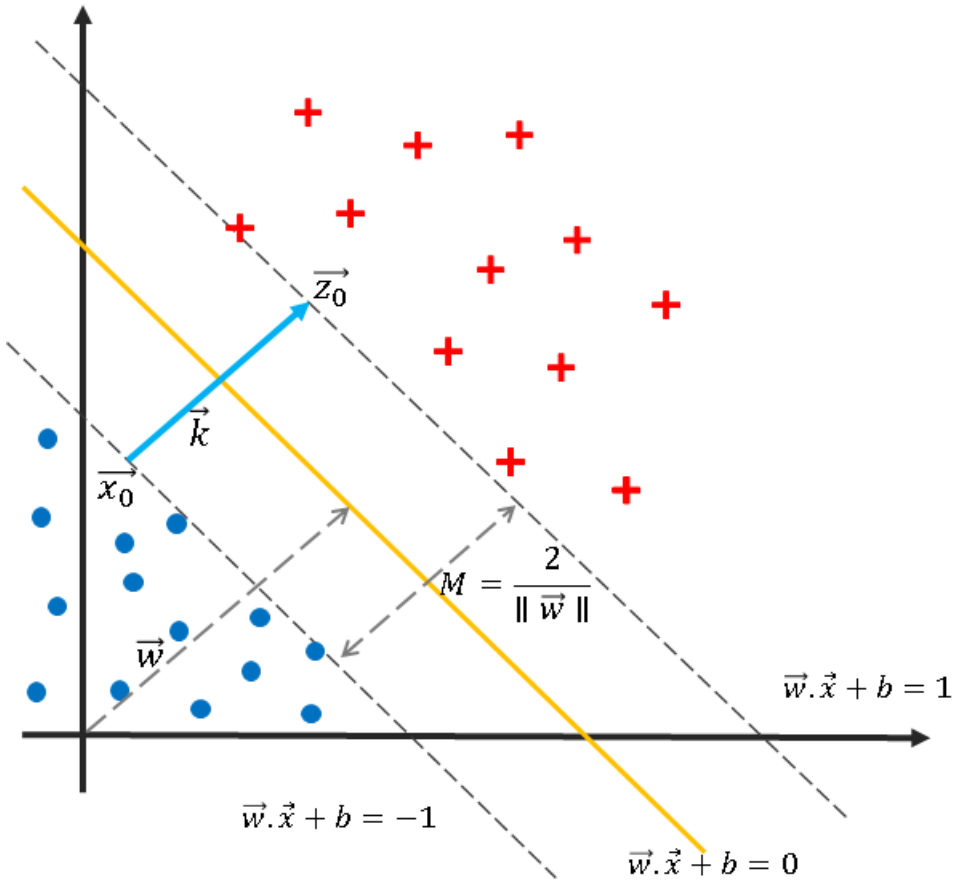


Figure 2.1: *Hyperplane in 2D, $y = \vec{w}.\vec{x} + b$ [8]. Yellow line is the hyperplane classifying the data points where as dotted lines parallel to it are the support hyperplanes*

So, the optimization problem becomes,

$$\begin{aligned} &minimize \quad \frac{1}{2}||\vec{w}||^2 \\ &subject\ to \quad y_i(\vec{w}.\vec{x_i} + b) \geq 1, \forall i \end{aligned} \tag{2.5}$$

This implies that we are assuming that classes are *perfectly-separable*. This is special case in SVM known as **hard-margin SVM** or **Vanilla SVM**. But, in real life this is not always the case. We allow some points to lie within other boundary. This is called **soft-margin SVM**. Also, sometimes data is not linearly classifiable. So, we apply *kernel trick*. It projects the points to higher dimensions so that the data is classified by a hyperplane[Fig. 2.2]. There are different types of kernels like *rbf, polynomial, linear*, etc.

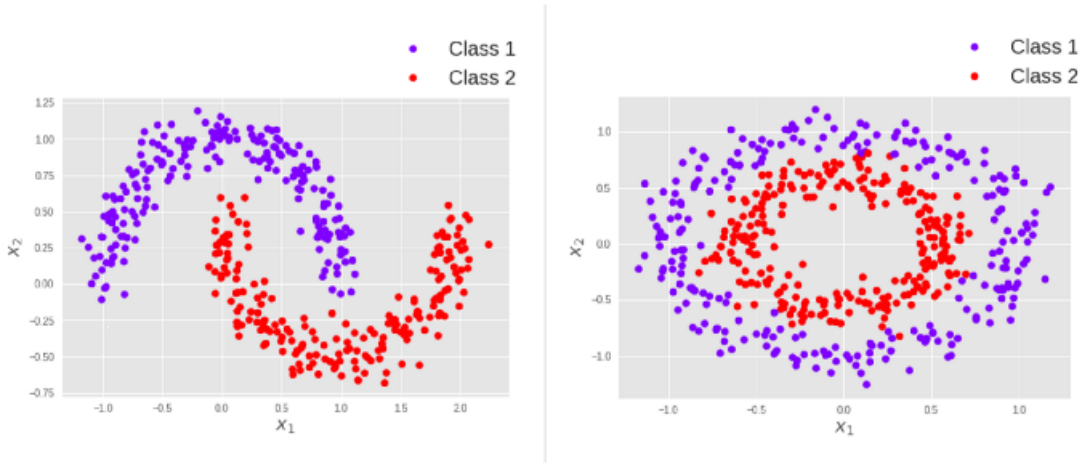


Figure 2.2: *As you can see, left image is not classifiable by a hyperplane. So, it is projected to 3D, so that is classifiable by a 2D plane*

So, the modified optimization problem becomes,

$$\begin{aligned} &\min_{w,b,\zeta} \ \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\zeta_i \\ &\textit{subject to} \ \ y_i(\vec{w}.\phi(\vec{x_i}) + b) \ \geq \ 1 - \zeta_i, \ \forall i, \ \zeta_i \geq 0 \end{aligned} \tag{2.6}$$

where, ϕ is the higher dimensional mapping function and ζ_i is the maximum allowable distance from the correct decision boundary for the samples
The dual form of this problem is

$$\begin{aligned} &\min_{\alpha} \ \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ &\textit{subject to} \ \ y^T\alpha = 0, \ 0 \leq \alpha_i \leq C, \ \forall \ i \end{aligned} \tag{2.7}$$

where, α_is are the dual coefficients, $Q \ = \ y^T K(x,x)y$, K being the kernel and e is the 1D vector of 1s

Now, class prediction for a sample x becomes,

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) - b \tag{2.8}$$

where, SV is the set of support vectors and b can be obtained by finding a support vector (x_j, y_j) on the decision boundary [9],

$$b = \vec{w}.\vec{x} - y_j \tag{2.9}$$

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x_i} \tag{2.10}$$

Chapter 3

Non-convex optimization framework

We form a new objective function by combining the standard SVM loss and the discrepancy problem and optimize it

Using eqns 3.3, 3.4, 3.7 and 3.8, the new objective function becomes,

$$(1-\alpha)\{\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i K(x_i,x_j) c_i y_j - \sum_{i=1}^{n} c_i\} + \alpha\{\max_{S_i \in S} \left| \sum_{x_j \in S_i} sgn(\,[\sum_{i \in SV} c_i y_i K(x_i,x_j)]\, - b) \right|\}$$

$$\sum_{i=1}^{n} c_i y_i \;=\; 0 \;\forall i$$

$$0 \;\leq\; c_i \;\leq\; \frac{1}{2n\lambda} \;\forall i \tag{3.1}$$

where the terms c_i are the dual coefficients and b can be obtained from eqn. 3.9

In this project, for prediction, instead of taking summation over the support vectors, we take summation over all data, as anyway c_i will be zero(negligible during training) for correct class predictions. So, eqn 3.8 becomes,

$$\sum_{i \in X} y_i c_i K(x_i,x_j) - b \tag{3.2}$$

Similarly, for calculating b, we take summation over all the input vectors and finally take the average of it. So, b becomes,

$$\underset{i}{avg}(\sum_{j=1}^{n} y_i c_i K(x_i, x_j) - y_j) \tag{3.3}$$

Finally substituting eqns 4.2 and 4.3 in 4.1,

$$\begin{aligned} (1-\alpha)\{\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i K(x_i, x_j) c_i y_j - \sum_{i=1}^{n} c_i\} + \\ \alpha\{\max_{S_i \in S}\left|\sum_{x_j \in S_i} sgn(\,[\sum_{i=1}^{n} c_i y_i K(x_i, x_j)] \; - \underset{i}{avg}(\sum_{k=1}^{n} y_i c_i K(x_i, x_k) - y_k)\,)\right|\} \\ \sum_{i=1}^{n} c_i y_i \;=\; 0 \;\forall i \\ 0 \;\leq\; c_i \;\leq\; \frac{1}{2n\lambda} \;\forall i \end{aligned} \tag{3.4}$$

Chapter 4

Results

We have used adult data [10] for the classification. Below are the results from running *sklearn's* SVM, MLP classifier, *zafar et al.* fairness algorithm [11] and finally, our algorithm on the data set.

Table 4.1 shows the original p-rules of the data. The first element in the braces adjacent to p-rules represent fraction of non-protected(here *male*) people in the positive class(salary $\geq$ 50000) and second element represents fraction of protected people in the positive class. As you can see, the data set is clearly biased towards non-protected people

data size	**p-rule(non-pro in pos, pro in pos)**	**p-rule training**	**p-rule test**
10000	36(31,11)	38(31,12)	33(32,10)
13913	38(31,12)	38(30,11)	38(32,12)
17827	39(30,12)	37(30,11)	44(30,13)
21740	36(32,11)	36(32,11)	37(31,11)
25654	37(31,12)	37(31,11)	38(31,12)
29567	36(31,11)	36(31,11)	37(31,11)
33481	36(31,11)	37(31,12)	34(31,11)
37394	37(31,11)	37(32,12)	36(31,11)
41308	37(31,11)	37(31,12)	36(31,11)
45222	36(31,11)	36(31,11)	36(32,11)

Table 4.1: original p-rules

In case of SVM(Table 4.2), accuracy(both train and test) and p-rules are almost same i.e around 80 and 37 respectively over the full data set except for the sudden drop

in p-rule for a data-size of 13913. % of non-protected people in positive class dropped by approx. 20%, with an average of 10, whereas % of protected people in positive class dropped by just 6%, with an average of 5. Fig. 4.1 shows accuracy *vs* data size plot for SVM

data size	train accuracy	test accuracy	p-rule training	p-rule test
10000	0.80	0.80	49(10,5)	54(11,6)
13913	0.80	0.78	44(6,2)	44(5,2)
17827	0.80	0.80	55(10,6)	58(11,6)
21740	0.80	0.79	52(11,6)	50(11,5)
25654	0.80	0.80	54(11,6)	55(11,6)
29567	0.79	0.80	50(11,5)	48(11,5)
33481	0.79	0.80	52(11,6)	50(11,5)
37394	0.79	0.80	49(11,5)	58(10,6)
41308	0.80	0.80	51(11,6)	54(11,6)
45222	0.80	0.80	52(11,6)	50(11,6)

Table 4.2: *sklearn* SVM classifier with *rbf* kernel

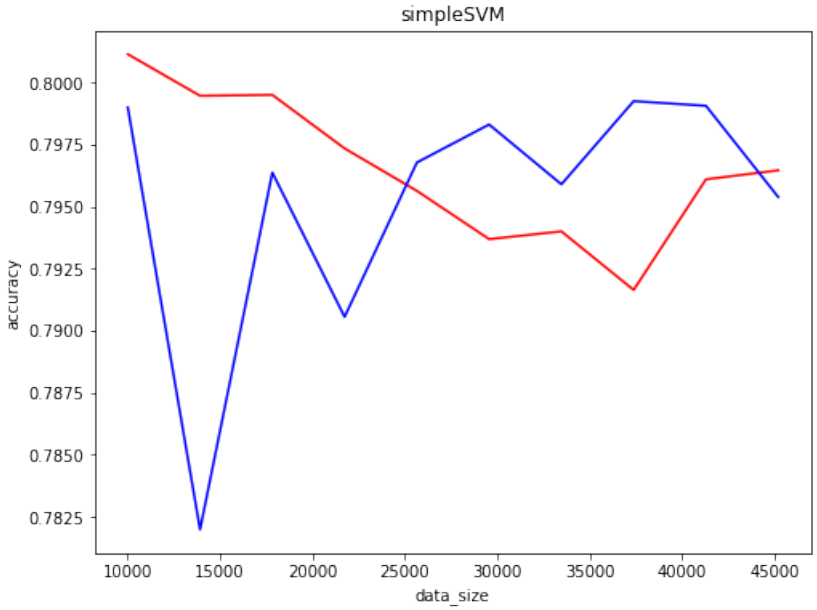


Figure 4.1: *SVM accuracy over different data sizes*

In case of MLP(Table 4.3), accuracy(both train and test) is almost same i.e around 84, over the full data set where as p-rules ranged over 30-38 with an average of 33. % of non-protected people in positive class dropped by an average of 5% whereas % of protected people in positive class dropped by just 2-3%. Fig. 4.2 shows accuracy *vs* data size plot for SVM

data size	train accuracy	test accuracy	p-rule training	p-rule test
10000	0.86	0.84	33(24,8)	30(25,7)
13913	0.83	0.82	38(16,6)	38(16,6)
17827	0.86	0.84	36(25,9)	37(28,10)
21740	0.84	0.83	30(22,7)	29(22,6)
25654	0.85	0.85	34(26,9)	35(26,9)
29567	0.85	0.82	32(25,8)	32(26,8)
33481	0.85	0.85	33(26,8)	31(26,8)
37394	0.86	0.85	33(30,10)	32(30,9)
41308	0.85	0.84	36(29,11)	36(28,10)
45222	0.85	0.85	31(25,8)	30(26,8)

Table 4.3: *sklearn* MLP classifier

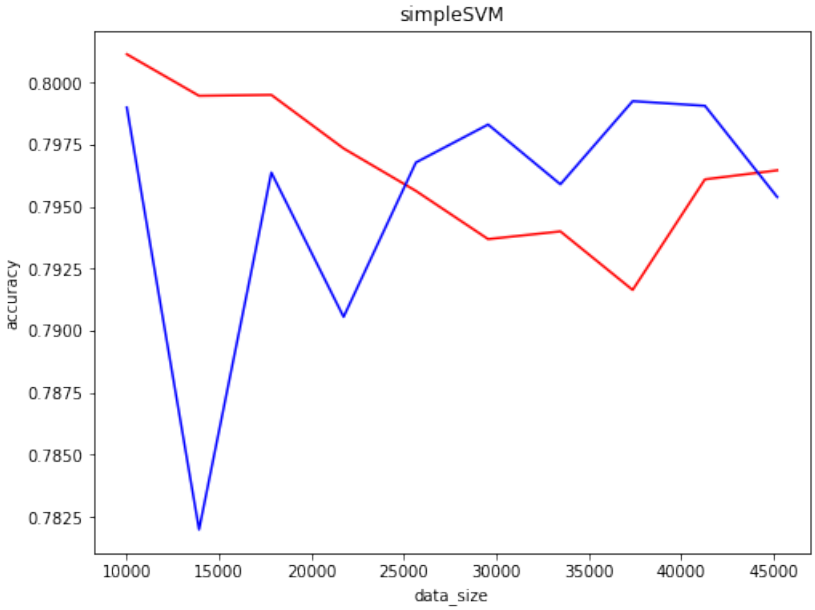


Figure 4.2: *MLP accuracy over different data sizes*

We have run *Zafar et al.* in two ways: Constrained and Unconstrained. In case of unconstrained classifier(Table 4.4), accuracy(both train and test) is almost same i.e around 0.84 for all data sizes where as p-rule showed variation.

data size	train accuracy	test accuracy	p-rule training	p-rule test
10000	0.85	0.84	38(26,10)	34(26,9)
13913	0.85	0.85	35(26,9)	37(24,9)
17827	0.85	0.84	34(26,9)	29(25,7)
21740	0.85	0.85	33(25,8)	32(25,8)
25654	0.84	0.84	77(19,14)	79(19,15)
29567	0.84	0.83	76(19,14)	77(19,15)
33481	0.84	0.83	78(19,15)	72(19,14)
37394	0.85	0.84	33(26,8)	32(26,9)
41308	0.83	0.84	79(19,15)	33(27,9)
45222	0.85	0.85	32(26,8)	34(26,9)

Table 4.4: *Zafar's* Unconstrained classifier

data size	train accuracy	test accuracy	p-rule training	p-rule test
10000	0.84	0.83	77(19,14)	85(18,15)
13913	0.84	0.83	76(19,15)	96(16,16)
17827	0.84	0.83	76(19,15)	80(18,15)
21740	0.84	0.84	77(18,14)	83(19,16)
25654	0.85	0.84	34(26,9)	79(19,15)
29567	0.86	0.85	33(28,9)	35(28,10)
33481	0.84	0.84	32(25,8)	32(24,8)
37394	0.84	0.83	79(19,15)	79(19,15)
41308	0.85	0.83	32(26,8)	79(19,15)
45222	0.84	0.83	77(19,15)	79(19,15)

Table 4.5: *Zafar's* classifier with fairness constraint

In case of constrained classifier(Table 4.5), accuracy(both train and test) is almost same i.e around 0.84 for all data sizes where as p-rule showed variations in the middle, but, finally saturated at 79%

Fig. 4.3 shows p-rule test plots over diff data sizes for unconstrained and constrained

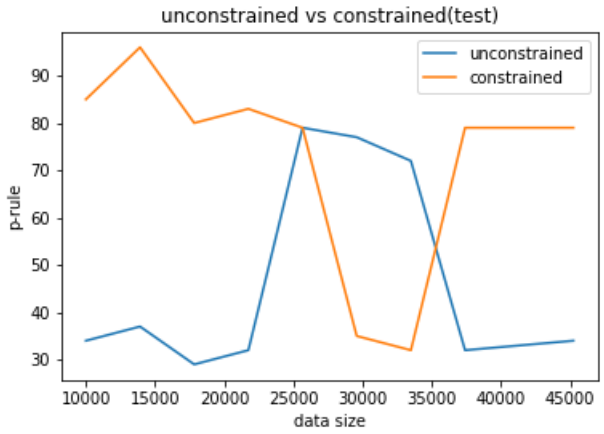


Figure 4.3: *p-rule over different data sizes*

We have used scipy's COBYLA optimizer to optimize our loss function. We have also run other optimizers like scipy's SLSQP, Powell, Nelder-Mead, etc. but they were very slow and highly inaccurate. You can find the results of COBYLA optimizer below

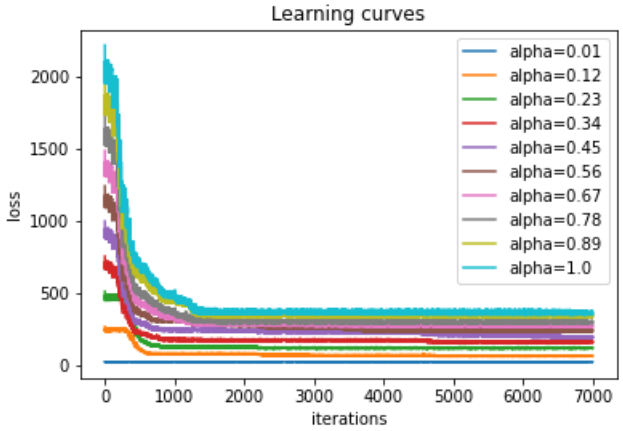


Figure 4.4: *Learning curve*

As you can see figures 4.4, 4.5a, 4.5b, as α increases loss increases and accuracy decreases, which exactly matches with the theory. Similarly, it applies to fraction of protected and non-protected people in positive class.

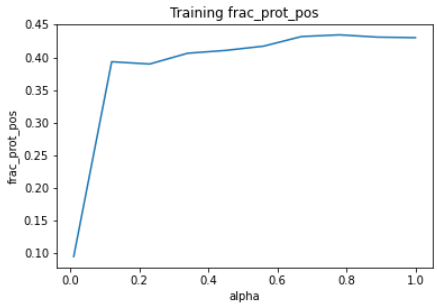
(a) *(b)*

Figure 4.5: *Accuracies*

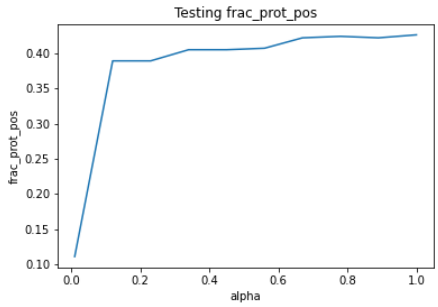
(a) *(b)*

Figure 4.6: *% of non-protected people in positive class*

In figs. 4.8a and 4.8b you can find the plots of p-rules *vs* alpha. As evident from the values of % of people in positive class(*figs.* 4.6a, 4.6b, 4.7a and 4.7b), the p-rules are around 80-90% .

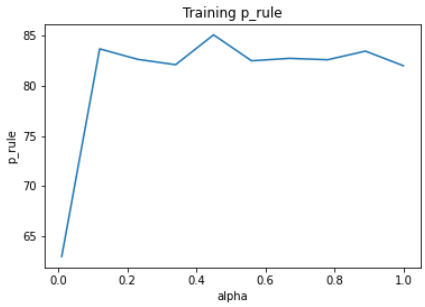


(a)

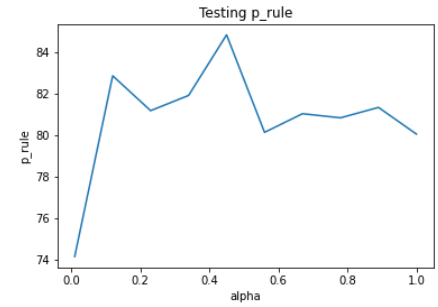


(b)

Figure 4.7: *% of protected people in positive class*



(a)



(b)

Figure 4.8: *p-rules*

Chapter 5

Conclusion and Future Work

In this project, we defined and implemented a notion of fairness based on discrepancy theory. This framework guarantees equal opportunity at the cost of accuracy. Comparing to other algorithms mentioned, it provides a far better *equal opportunity* at the cost of accuracy. α around 0.1 gives around 70% accuracy and a decent positive class proportion around 40-50%. Where as the postive class proportion of both non-protected and protected people were atmost 38% for other classifiers. Even the p-rule is around 80-85% for *alpha* around 0.1. To make it more robust, calibration after the classification can be implemented. Although, we need to verify this on other data sets.

There is a huge scope for improvements, but, requires some theoretical study. All the non-smooth functions in the objective function can be replaced with *continuous, differentiable* functions. For example, modulus operation can be replaced with a *square* operation, *signum* function can be replaced with *tanh*. MinMax in discrepancy problem can be replaced with minimizing the average discrepancies over protected attribute values, making the objective function smoother which inturn makes the optimization easy. Non-convex optimizers can be used to optimize the objective function

Algorithmic bias is a pressing issue. It might cause serious problems to the companies and also the individuals. There has been a lot of accusations and allegations, primarily regarding the race, already. I firmly believe this is the right time we take certain measures like increasing research, speeding up inclusion of fairness implementations into the already existing softwares, adding courses to the curriculum, etc to tackle this. We hope our idea stimulates more research and study in this area and eventually help in solving this issue

References

[1] Safiya Umoja Noble. Algorithms of oppression, Jul 2019.

[2] Latanya Sweeney. Discrimination in online ad delivery, 2013.

[3] Adam Rose. Are face-detection cameras racist?, Jan 2010.

[4] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. How we analyzed the compas recidivism algorithm, May 2016.

[5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Randomness and Complexity. Cambridge University Press, 2000.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Suraj Donthi. Support vector machines, dual formulation, quadratic programming amp; sequential minimal optimization, Feb 2021.

[9] Wikipedia. Support-vector machine, May 2021.

[10] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[11] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification, 2017.