Student Project: A Framework to Simulate the Impact of Static Decision Rules on Long-Term Fairness

Danilo Brajovic *(383 5528)*
Tübingen University
danilo.brajovic@student.uni-tuebingen.de

Supervisor: Philipp Hennig, Tübingen University
Reviewer: Niki Kilbertus, MPI-IS, Tübingen

Abstract

Fair algorithms are designed to mitigate discrimination against certain demographic groups such as minorities and other disadvantaged groups in automatic decisions. Early fairness-aware machine-learning methods viewed fairness from a static point of view and did not consider delayed impacts of decisions in the future. Recently, long-term impacts and dynamics of decisions moved into the focus and the long-term effects of decisions have been studied. Some of the key insights are that decisions that appear to be fair from a static point of view may have negative long term impact. This student project develops a framework to simulate the influence of static decision rules on long-term fairness under two assumptions regarding the dynamics of the data generation. Either decisions effect the whole group of people sharing a protected attribute, or only the individual itself is affected by a decision. Depending on the assumption about the underlying data generation mechanism, the same decision rule can have different long-term impact. The framework is designed to work with the AIF360[1] toolkit in order to provide out-of-the-box access to fairness metrics and algorithms.

Introduction

The basic setup for this project is a binary classification setting with a matrix of features $X \in \mathbb{R}^{n\times m}$, a vector of true labels $y \in \{0,1\}^n$ and a vector of protected attributes $a \in \{0,1\}^n$ (such as gender or ethnicity) of n individuals. A fairness aware decision maker (sometimes called agent) tries to learn a decision function $d : \mathbb{R}^m \times \{0,1\} \to \{0,1\}, (X_i, a_i) \mapsto \hat{y}_i$ that maps the features for each individual to binary decisions. The decision function has to satisfy both the agents utility $u : \{0,1\} \times \{0,1\} \to \mathbb{R}, (y_i, \hat{y}_i) \to u(\hat{y}_i, y_i)$ (usually the accuracy) as well as a fairness constraint measuring fairness violations on the overall set of predictions for all individuals:

$$f : \mathbb{R}^{n\times m} \times \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \mathbb{R},$$
$$(X, a, \hat{y}, y) \to f(X, a, \hat{y}, y)$$

Examples for fairness constraints are, for instance, error based constraints that require equal error rates between different groups of the protected attribute. The objective of the agent is to find an optimal decision function d^* satisfying a tradeoff between utility and fairness:

$$d^* = \max_{\hat{y}} \{\lambda u + (1-\lambda) f\} \tag{1}$$

[1] https://aif360.mybluemix.net/

Static Fairness in this project will refer to fairness metrics f or decision functions d that do not care about previous or future impact in the decision process. This will be useful when we extend the setup to discrete time steps t later. In particular, we assume that decisions at time t change the distribution X, y are sampled from in the next time step $t+1$. A static decision function does not consider such possible impacts.

Dynamic Fairness

In dynamic settings, a decision is considered to have an impact on the future. Several dynamics have been pointed out recently. If a university accepts an applicant based on the qualification, this will increase the qualification of this applicant again. If a decision maker rejects individuals too often, these applicants may retreat from applying again [1]. Feedback loops strengthen inequality; when a certain group is known to be criminal the police places more attention to this group and automatically identifies more crimes and labels new individuals as criminal. Being labeled a criminal again decreases chances for this person and enforces criminal behavior [2].

We are typically lacking longitudinal dataset required to study dynamic effects. Therefore, most works make assumptions about the underlining data generation and simulate or compute the results analytically. This project considers cases where a decision has an impact on the features of individuals and provides a framework to simulate the results. This is for instance the case in the university example or for a company hiring people. An individual who was hired may increase their qualification by experience after some time, even if this individual was not qualified for the job at first. These dynamics are implemented as a sequential data generator into a framework to simulate effects of static decision rules on dynamic data. Two different assumptions about the data generation are implemented. Either whole groups benefit from positive decisions, or only the individuals affected by the decision benefit. The motivation for these two can be seen as grounded in the idea of social learning [3, 4]. The social environment serves as a role model. If an individual from a protected group is given an opportunity, for example being accepted for university, this might motivate other individuals from the same group. However, this influence can differ. Our framework considers marginal cases where either the whole group or only individuals benefit and visualizes that the same decision rule will have different impacts on the long-term fairness under those two assumptions.

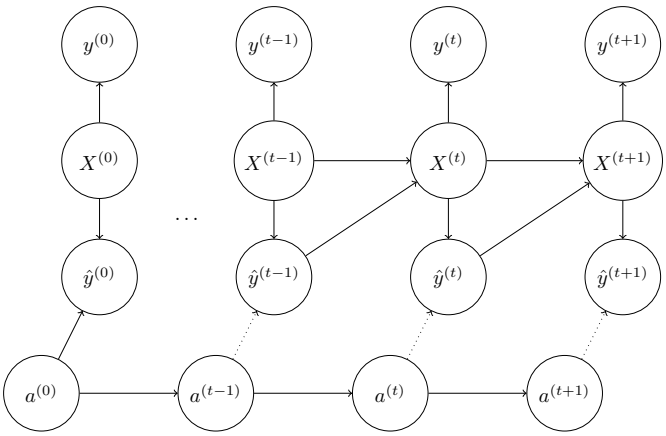

Figure 1: The features X may influence the true labels y. The decision maker is not marked in the figure, but makes predictions $\hat{y}$ from features X. Therefore, X also influences $\hat{y}$. The protected features a are constant over time. Features X between two time steps t should be similar. The protected attribute is only considered in the first time step (during data initialization). It can be considered by the decision maker later but does not have to. This is indicated by the dotted lines. What should emerge from the graph is that the protected features a have an impact on the true labels y via predictions $\hat{y}$ but become independent once conditioned on them; $y \perp\!\!\!\perp a \mid \hat{y}$. Even if the influence of a is removed at some point, the unfairness in the data will be present.

Problems with Dynamics and Static Fairness An often-discussed drawback of static fairness definitions is their failure to distinguish causal relationships [5]. Without any additional knowledge, a static decision rule cannot distinguish whether an attribute is causal for an outcome or only serves as proxy. For example, whether gender defines the qualification for a job or whether it is only correlated due to a common cause.

This is a similar problem addressed here. In the long term setup, the causal question is whether the environment was impacted by previous decisions and what the impact of a decision will be on the future. Note that the static decision functions considered in this framework do not try to answer this and do not reason about their impact. Causal fairness definitions in contrast ask *why* a certain decision was made. In that sense, a dynamic decision rule or long-term fairness metric (which is not discussed in this project) could be considered as sub type of causal fairness definitions since it asks *why* the environment state is as it is and what *will* be the impact of a decision. The graph in Figure 1 describes the data generation assumed.

Contributions

There are three main contributions implemented in this project. A data generator provides sequential data given old predictions at each time step for two different data generation assumptions. A plot generator runs a loop repeatedly sampling new data, evaluating metrics and finally plotting the results. Wrapper classes provide interfaces to the AIF360 toolkit to provide out-of-the-box access to numerous metrics and fairness aware algorithms. Code contributions are described in jupyter-notebooks available on GitHub[2]. This report describes the basics and motivation of the data generation.

Related Work

Several works have proposed models to explain different dynamic scenarios. Some of them include feedback loops in predictive policing [2], dynamics in the labor market where positive decisions improve the reputation of the whole group [6], lending scenarios where loan decisions have an impact on groups [7, 8], and scenarios where individuals try to trick the decision maker [9]. Recently, Google published a framework called *ML-fairness-gym* to simulate some environments and train reinforcement learning agents [10]. The difference between these environments and the one proposed here is that our framework considers two types of different dynamics.

Finally, the difference between our framework and ML-fairness-gym is that the former is designed with the goal to train reinforcement learning agents, which reason about their future impact, while this framework aims to offer an easy way to estimate effects of simple static decision rules already available in the AIF360 toolkit.

Methods

Setup and Data Generation

The setup is a simple repeated binary classification setting at discrete time steps t. The superscript $^{(t)}$ is used to differentiate the time steps. An agent observes data $X^{(t)}, a^{(t)}$ of n individuals and makes decisions $\hat{y}_i^{(t)} = d(X_i^{(t)}, a_i^{(t)})$ at each time step t. The environment then computes some fairness metrics f based on the decisions and returns a plot after the final iteration. As a result of the decision $\hat{y}^{(t)}$ the features $X^{(t+1)}$ of the next time step are influenced.

The variables $y^{(t)}, a^{(t)}, \hat{y}^{(t)} \in \{0,1\}^n$ are binary vectors for simplicity. Since the goal of this project is visualization, the features $X^{(t)} \in \mathbb{R}^{n\times 2}$ are two dimensional. The number of individuals n is constant over time. That is, the agent observes the same individuals in each time step. Each row of $X^{(t)}$ refers to the features of one of the n individuals denoted by the subscript $_i$. Therefore, $X_i^{(t)} \in \mathbb{R}^2, y_i^{(t)}, a_i^{(t)}, \hat{y}_i^{(t)} \in \{0,1\}$ are the features, true label, protected feature and decision for individual i at time t. The agent itself a simple classifier that might be constrained w.r.t some fairness measure f as described in equation 1 and is denoted as d for 'decision function'. We consider $\hat{y}_i^{(t)} = 1$ to be the beneficial decision and $a_i^{(t)} = 1$ the privileged group.

A simple example for such an environment could be a loan environment similar to the one in ml-fairness-gym. The agent is a bank accepting or rejecting loan applications. $X^{(t)}$ would be features of n individuals the bank observes and uses to decide whether to accept or reject a loan at time t. $y_i^{(t)}$ defines whether individual i will default on the loan and $\hat{y}_i^{(t)}$ whether individual i was granted a loan. Individuals repeatedly apply for loans in each time step and change their features $X_i^{(t+1)}$ of the next time step $t+1$ as response to a decision $\hat{y}_i^{(t)}$ at time t.

[2] https://github.com/danilobr94/dynamic-aif-framework

Dynamics are implemented by assuming that the probability for a positive label at time step t is proportional to the number of positive predictions in the past. The positive label is 1. Therefore, the probability for a positive label for individual i at time t should be proportional to:

$$P(y_i^{(t)} = 1) \propto \sum_{j \in G} \sum_{k=1}^{T} \hat{y}_j^{(t-k)} \tag{2}$$

G defines the group of individuals influencing individual i and is explained later in more detail. T is the number of previous time steps considered. The data generator is implemented by initially sampling the unprotected features $X_i^{(0)}$ for each individual from one of two different bivariate Gaussians. Individuals with a positive label are sampled from a Gaussian with mean $\mu_{pos} \in \mathbb{R}^2$ and negative labeled individuals from a Gaussian with mean $\mu_{neg} \in \mathbb{R}^2$.

The protected attribute a is constant over time and only sampled on initialization. To account for discrimination in the initial data generation step, more individuals from the protected group are sampled from the negative cluster, but the data generator does not assume an actual causality between the protected feature and the label, that is $y \perp\!\!\!\perp a \mid \hat{y}$. In summary, four blobs of data are sampled on initialization:

(i) Individuals with positive label and privileged group from $\mathcal{N}(X;\ \mu_{pos}, \Sigma_{pos})$ with $y = 1$ and $a = 1$.

(ii) Negative label and unprivileged group from $\mathcal{N}(X;\ \mu_{neg}, \Sigma_{neg})$ with $y = 0$ and $a = 0$.

(iii) Positive label and unprivileged group from $\mathcal{N}(X;\ \mu_{pos}, \Sigma_{pos})$ with $y = 1$ and $a = 0$.

(iv) Negative label and privileged group from $\mathcal{N}(X;\ \mu_{neg}, \Sigma_{neg})$ with $y = 0$ and $a = 1$.

The bias only comes from the initial number of individuals per group. Most individuals are sampled from group (i) and the least from group (iv). Similarity of features in the dynamic steps is achieved by sampling new points with variance Σ_{local} around the old points. The relationship between the variables is displayed in Figure 1. The pseudo code for this step is:

Data: $X^{(t-1)}, y^{(t-1)}, a^{(t-1)}, \Sigma_{local}, \alpha$

$X^{(t)}, y^{(t)} = [], [];$
for *each individual* i *in* $X^{(t-1)}$ **do**
  Compute **offset**$_i$;
  $X_i^{(t)} \sim \mathcal{N}(X_i^{(t-1)} + \textbf{offset}_i,\ \Sigma_{local})$;
  $y_i^{(t)} \leftarrow$ Label of μ_{pos} or μ_{neg} closer to $X_i^{(t)}$;
end

Result: $X^{(t)}, y^{(t)}, a^{(t-1)}$

To account for the dynamics in equation 2, each individual moves either towards the positive or negative cluster proportional to the sum of previous predictions according to an offset:

$$\textbf{offset}_i = \alpha \cdot \vec{v}_i \cdot \frac{1}{T \cdot |G_i|} \sum_{j \in G_i} \sum_{k=1}^{T} \hat{y}_j^{(t-k)} \tag{3}$$

Where:

- $\alpha \in \mathbb{R}$ is the step size.
- $\vec{v}_i \in \mathbb{R}^2$ is the direction vector pointing from the features of the individual to the corresponding cluster.
  I.e. $\vec{v}_i = \mu_{pos} - X_i^{(t-1)}$ or $\vec{v}_i = \mu_{neg} - X_i^{(t-1)}$ depending on whether $\sum_{j \in G_i} \sum_{k=1}^{T} \hat{y}_j^{(t-k)} > 0$ or not.
- G_i is the set of other individuals influencing individual i.
- $T \in \mathbb{R}$ is the number of previous steps considered.

G defines the assumptions about the data generation. The two implemented assumptions and their definition are:

Assumption 1: Positive decisions only affect individuals themselves. The probability for a positive label depends only on the predictions for the individual itself. For individual i this means $G_i = \{i\}$.

Assumption 2: Positive decisions affect the whole group sharing the protected attribute. The whole group benefits from positive decisions for individuals. For individual i this means $G_i = \{j | a_i^{(t)} = a_j^{(t)}\}$.

In general, any other subset is also possible. These two assumptions are provided because they capture two intuitively natural scenarios.

Code Contributions

So far, the motivation and implementation of the data generation was explained. Examples for the usage of the data generators is provided in jupyter-notebooks on GitHub. The other contributions of the framework are methods to simulate dynamics in a loop and interfaces for the methods from AIF360. Explanations for them are also provided on GitHub. In general, there are two data pipelines; the true and the baseline pipeline. In the true pipeline, data is generated based on the predictions of the decision rule and in the baseline one as if all previous predictions were positive. In the current implementation, the decision function is only trained on the true pipeline. A flag controls whether the function is retrained on the whole data set after each iteration or only trained once at the beginning. The pseudo code for the evaluation pipeline is given in the appendix. The next section provides examples of running the two data generators in the simulation framework.

Examples

In total, four experiments are discussed. An unconstrained and a constrained classifier is trained on both group and individual data generator. Scatter plots of the data points are provided in the appendix. Fairness is measured in terms of the base rate (the overall proportion of individuals with a positive label at each time step). This is not an optimal measure, but serves as good proxy for illustration purposes. We set $\mu_{pos} = [14, 12]^T$, $\mu_{neg} = [0, 1]^T$ and $\Sigma \approx I_2$ as covariance matrix for both clusters. We also set $\Sigma_{local} = 0.1 I_2$.

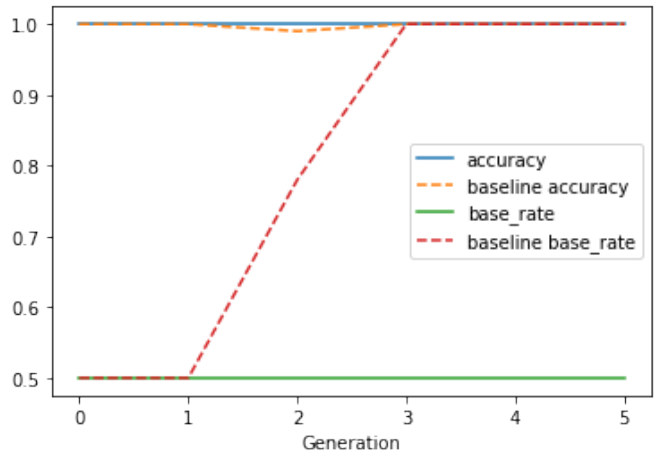

Figure 2: Five iterations of unconstrained logistic regression on individual generated data. The true data is constant in both accuracy and base rates. In the baseline data, the base rates increase constantly while the accuracy drops for a short time. The details can be seen in the scatter plot in Figure 6 of the appendix.

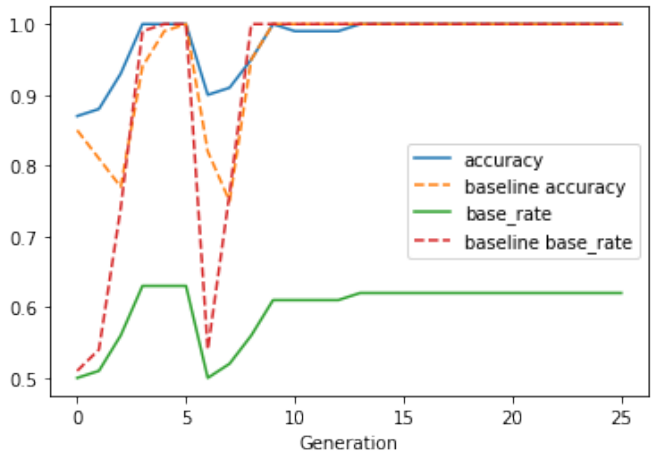

Figure 3: 25 iterations of an artificial-constrained decision function on individually generated data. The base rate slightly increases in the true data while the accuracy slightly drops. In the baseline data accuracy and base rate toggle for a few iterations but converge towards an optimal value of 1.0 afterwards. The toggling happens because the learned decision boundary is almost orthogonal to the true boundary and points randomly cross it until they converge to the positive cluster. This is visible in the scatter plots displayed in Figure 7 of the appendix again.

Examples on Individual Data Generator

In the individual data generator, positive decisions have an impact only on the people affected by a decision.

Unconstrained Decision Rule Figure 2 shows the final plot after 5 iterations of an unconstrained logistic regression trained on the individual data. The points are scattered in Figure 6 of the appendix. In the true data, base rate and accuracy are constant over time. This is because the points are perfectly separable. Therefore, no points will change between clusters and the base rate stays constant. In contrast, in the baseline pipeline the base rate continuously increases until it reaches a value of 1.0 (all individuals moved to the positive cluster). The accuracy drops for a short period of time when points reach the area where learned decision function and the data generating decision function are misaligned (see Figure 6 in the appendix for details).

Constrained Decision Rule In Figure 3 an artificial fair-decision rule is applied to the individual generated data. Now, the fair-decision rule missclassifies some of the negative labeled individuals and thereby 'gives them a chance'. Therefore, the base rate slightly improves in the true data because the individuals who were given a chance improve their skills while the accuracy drops for s short period of time. In the baseline data, accuracy and base rates toggle around until they reach the perfect state (accuracy and base rate 1.0) after approximately 10 iterations. The toggling happens because the learned decision boundary is almost orthogonal to the true boundary and points randomly cross it until they converge to the positive cluster as visible in Figure 7 of the appendix.

Conclusion for Individual Data Generation

These two plots show that a perfect decision rule might not be desirable under the assumption that individuals benefit from positive decisions. Such decision rules would often be considered as fair from a static point of view, for example, if fairness is considered with respect to error rates. A perfect decision function stabilizes the inequality in the data although the model does not assume a causality between protected feature and qualification. The correlation is introduced only by a skewed decision in the first decision step (during data generation in this model) and enforced by the decision function. The constrained decision rule produces better results since it slightly improves the base rate. However, it only improves the situation for those individuals who were lucky and assigned a positive label by the fair decision rule. Compared to the baseline data, the base rate in the true data could improve. All in all, the constrained decision rule produces better results when measuring the base rate of positive-labeled individuals. It still fails to achieve optimal results as produced by the baseline pipeline.

Examples on Group Data Generator

In the group data generator, a decision influences the whole group sharing a protected attribute. Only positive decisions have an impact on the group.

Unconstrained Decision Rule The results for group data generation without constraint are displayed in Figure 4. This time, base rates reach optimal values of 1.0 for both the baseline data and the true data, although it takes longer in the true data. This is because only positive decisions influence the outcome. Hence, individuals from both groups of the protected feature move towards the positive cluster. However, those of the unprotected group move faster because there are more 'role model' points with a positive label.

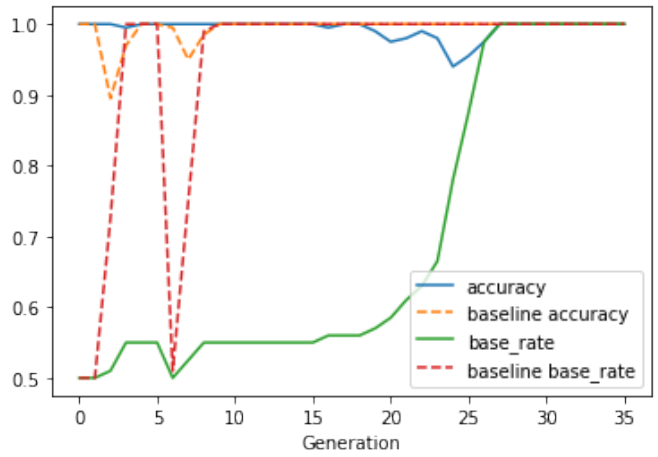

Figure 4: 35 iterations of an unconstrained linear support vector machine trained on groupwise-generated data. Base rates increase to 1.0 in both baseline and true data. The true data, however, needs approximately 20 iterations to achieve this while the baseline data needs only 5. Details are discussed in Figure 8 of the appendix.

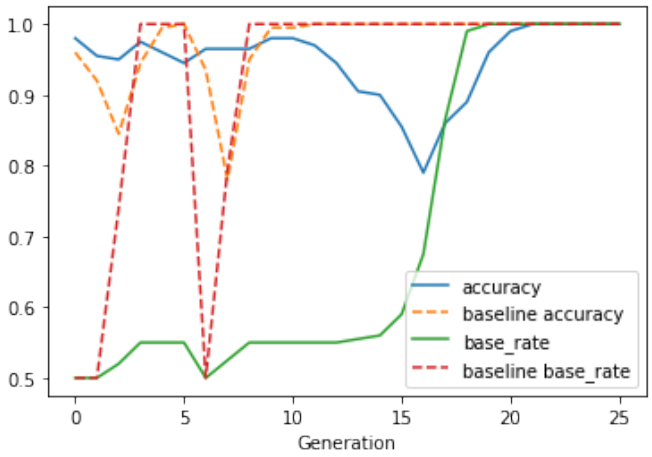

Figure 5: 25 iterations of an artificial constrained decision function on groupwise-generated data. Results are similar to the unconstrained data, but the true pipeline now reaches a base rate of 1.0 in 15 iterations instead of 20 due to the larger number of 'role models'. The accuracy decrease is larger, because the fair decision rule is almost orthogonal to the data generating boundary. Details are shown in Figure 9 of the appendix.

Constrained Decision Rule Results with a constrained decision function are displayed in Figure 5. Results are similar to the previous ones. Both pipelines converge towards perfect values of 1.0 for all metrics. However, it is faster in both cases since more individuals are assigned a positive value. This time there is more oscillation and the decrease in accuracy in the true data pipeline is larger.

Conclusion

In the group data generation, all decision rules converge towards optimal values in the long term. This is because only positive decisions have an impact on the data and all individuals will move towards the positive cluster in the end. The difference is only how fast this happens. This behavior is not optimal. It would be more realistic if negative decision had an impact as well. This way, the base rates would be stable with a perfect decision rule.

Compared to the individual data generation, the group data generation produces optimal results in the long term. Hence, the long term effect of a decision rule is highly dependent on the underlying data generation assumption.

Discussion

The goal of this project is to provide a simple framework to explore and visualize the effects of different static decision rules under dynamic settings. The main statement is that the same decision rule can have different impact depending on the underlying data dynamics. In this project, only positive decisions had an impact on the dynamics. This is because the negative label was 0 and did not count negative in the sum in equation 2. This is not realistic and would change the results again. An improved data generation scheme with more considerations is therefore discussed in the next section.

Next Steps

Advanced Data Generation

One assumption in the data generators is that every individual benefits from positive decisions in the same way. First, this need not always be the case and, second, negative decisions should also have an impact on the dynamics. This could be modeled using a more complex framework with a hidden variable h representing the level of capability of each individual. Individuals would then benefit from positive decisions depending on the level of their hidden capability h. Furthermore, the distance in feature space to individuals with positive prediction should define how much someone benefits from the decisions of others. Closer points would benefit more than points further away. This way both assumptions implemented here could be integrated into one model by changing the distance at which points have an influence. The generator should be as extensive as possible to easily account for new dynamics and incorporate randomness.

The goal of the decision maker in this case could be to achieve high accuracy in short-term decision making and learn the hidden variable h in the long term in order to maximize the pool of qualified individuals in the future effectively.

References

[1] Tatsunori B. Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization, 2018.

[2] Danielle Ensign, Sorelle A. Friedler, Scott Neville, Carlos Scheidegger, and Suresh Venkatasubramanian. Runaway feedback loops in predictive policing.

[3] Hoda Heidari, Vedant Nanda, and Krishna P. Gummadi. On the long-term impact of algorithmic decision policies: Effort unfairness and feature segregation through social learning. *CoRR*, abs/1903.01209, 2019. URL http://arxiv.org/abs/1903.01209.

[4] tim boone, anthony j. reilly, and Marshall Sashkin. Social learning theory albert bandura englewood cliffs, n.j.: Prentice-hall, 1977. 247 pp., paperbound. *Group & Organization Studies*, 2(3):384–385, 1977. doi: 10.1177/105960117700200317. URL https://doi.org/10.1177/105960117700200317.

[5] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[6] Lily Hu and Yiling Chen. A short-term intervention for long-term fairness in the labor market. doi: 10.1145/3178876.3186044.

[7] Lydia T. Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. Delayed impact of fair machine learning.

[8] Benjamin Paaßen, Astrid Bunge, Carolin Hainke, Leon Sindelar, and Matthias Vogelsang. Dynamic fairness - breaking vicious cycles in automatic decision making. *CoRR*, abs/1902.00375, 2019. URL http://arxiv.org/abs/1902.00375.

[9] Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. The disparate effects of strategic manipulation. doi: 10.1145/3287560.3287597.

[10] Alexander D'Amour, Hansa Srinivasan, James Atwood, Pallavi Baljekar, D. Sculley, and Yoni Halpern. Fairness is not static: Deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 525–534, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3372878.

Appendix

Plot Generation Algorithm

Data: metric, generator, steps, clf, retrain-clf

$X_{old}, a_{old}, y_{old} \leftarrow$ generator.*initialize-data*;
clf.$fit(X_{old}, a_{old}, y_{old})$;

$\hat{y}_{old} \leftarrow$ clf.*predict*(X_{old}, a_{old});

$X_{old}^{base}, a_{old}^{base}, y_{old}^{base} \leftarrow X_{old}, a_{old}, y_{old}$;

TrueMetric.*append(metric*$(X_{old}, a_{old}, y_{old})$);
BaselineMetric.*append(metric*$(X_{base}, a_{base}, y_{base})$)

for *s in steps* **do**
  $X, a, y \leftarrow$ generator.*sample*$(X_{old}, a_{old}, \hat{y}_{old})$;
  TrueMetric.*append(metric(*X, a, y*))*;

  if *retrain-clf* **then**
    clf.*fit*$(X_{old}, a_{old}, y_{old})$
  end

  $\hat{y} \leftarrow$ clf.*predict(*X, a*)* ;
  $y_{pos} \leftarrow$ Positive Label of shape $\hat{y}$;

  $X^{base}, a^{base}, y^{base} \leftarrow$ generator.*sample*$(X_{old}^{base}, a_{old}^{base}, y_{pos})$;
  BaselineMetric.*append.(metric*$(X^{base}, a^{base}, y^{base})$*)*;

  X_{old}.*append*(X);
  a_{old}.*append*(a);
  y_{old}.*append*(y);
  $\hat{y}_{old}$.*append*$(\hat{y})$;

  X_{old}^{base}.*append*(X^{base});
  a_{old}^{base}.*append*(a^{base});
  y_{old}^{base}.*append*(y^{base});
end

Result: *Plot*(TrueMetric, BaselineMetric)

The green lines in the following plots are the learned decision boundaries by the classifiers. The border between white and red background is the data generating decision boundary. Points in the white area represent individuals with a positive label and points in the red area with negative label ($y = 1$ and $y = 0$). Green points were predicted with a positive label and red ones with a negative label ($\hat{y} = 1$ and $\hat{y} = 0$). The right column represents the baseline data pipeline and the left one the true pipeline.

Unconstrained Individual Data Generation

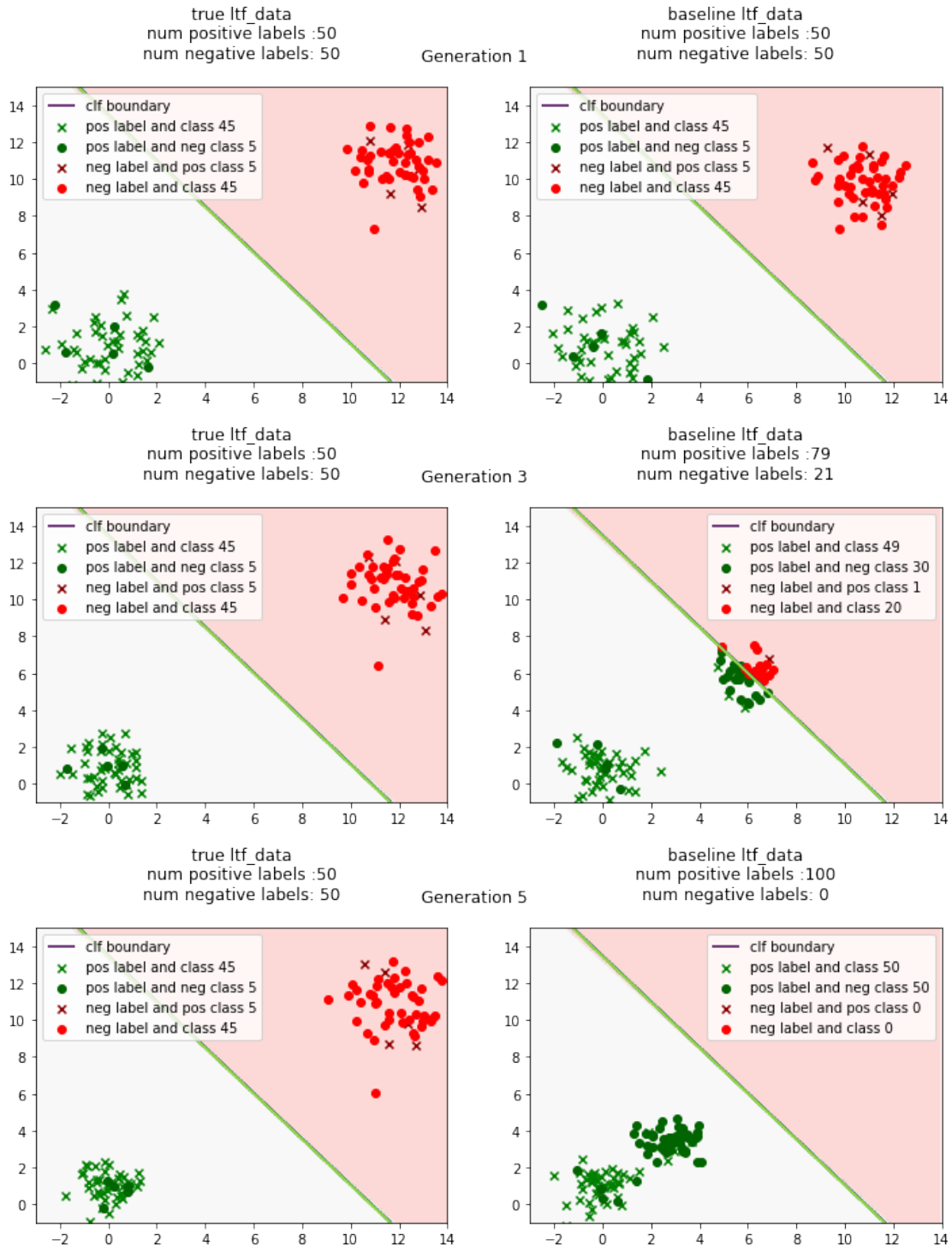

Figure 6: Unconstrained individual data generation. In the first generation, both baseline and true data are identical. The data points are perfectly separable and the learned decision boundary separates them without errors. Since there are no miss classifications, nothing changes in the true data for 5 generations. In the baseline data, all points start moving towards the positive cluster and cross the decision boundary within 5 iterations. The drop of accuracy visible in Fig. 2 happens when the red points reach the data-generating decision boundary but have not crossed the learned decision boundary as visible in generation 3.

Constrained Individual Data Generation

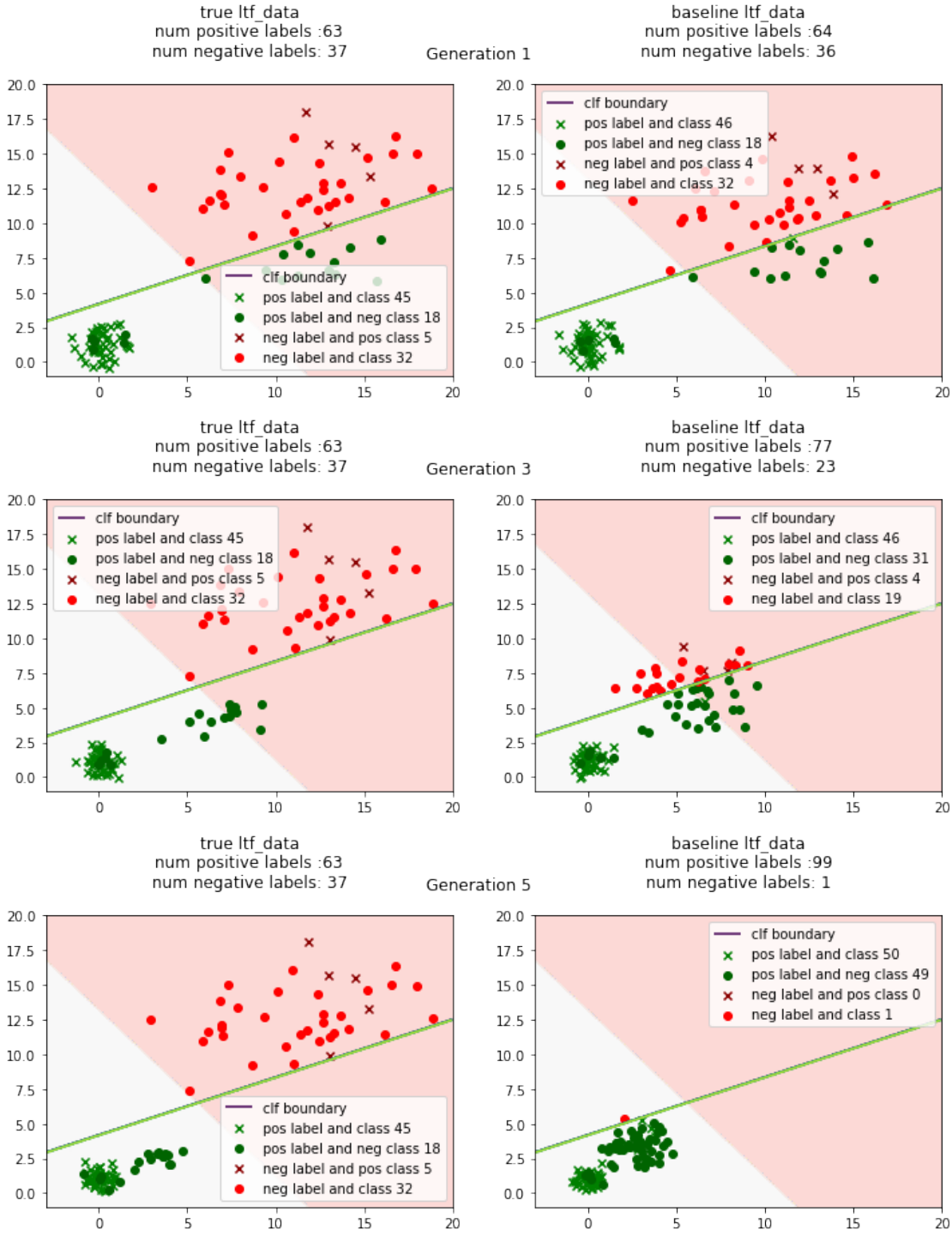

Figure 7: Five iterations of individual data generation with constrained decision function. The constrained decision rule is almost orthogonal to the true decision boundary. Therefore, some of the initially negative-labeled points are labeled positive. These points move towards the positive cluster in the true data. Results in the baseline data are the same as before, all points move towards the positive cluster. The oscillation in Figure 3 happens because the points in the baseline data move parallel to the decision rule and therefore randomly cross it sometimes.

Unconstrained Group Data Generation

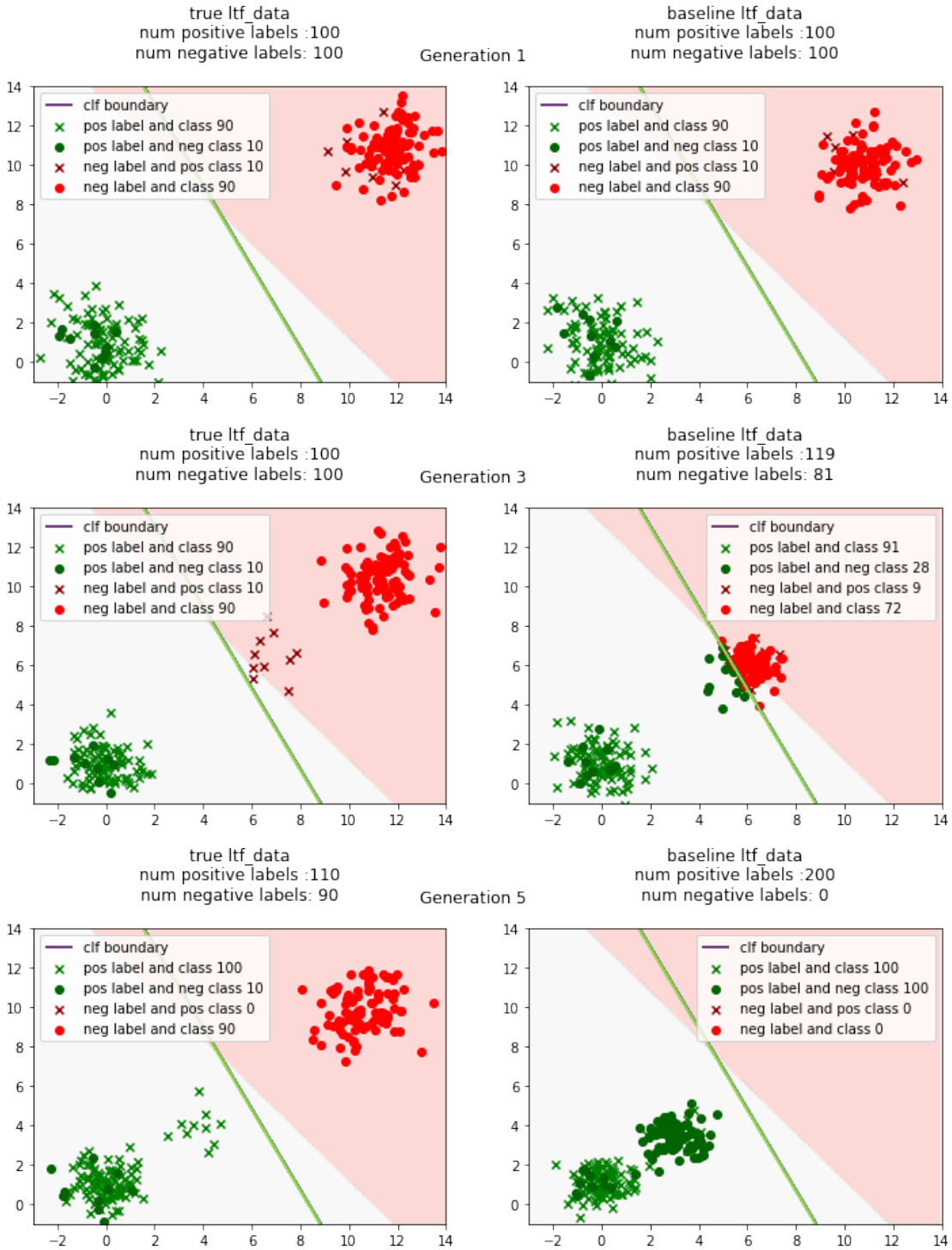

Figure 8: In the unconstrained data, a linear support vector machine is trained which results in a decision boundary further away from the true one. Again, all points move towards the positive cluster in the baseline data as a single blob. In the true data, the unprotected points move much faster, because there are more role models with a positive label. As visible in Figure 4, points in the true pipeline will also reach the positive cluster but it takes longer.

Constrained Group Data Generation

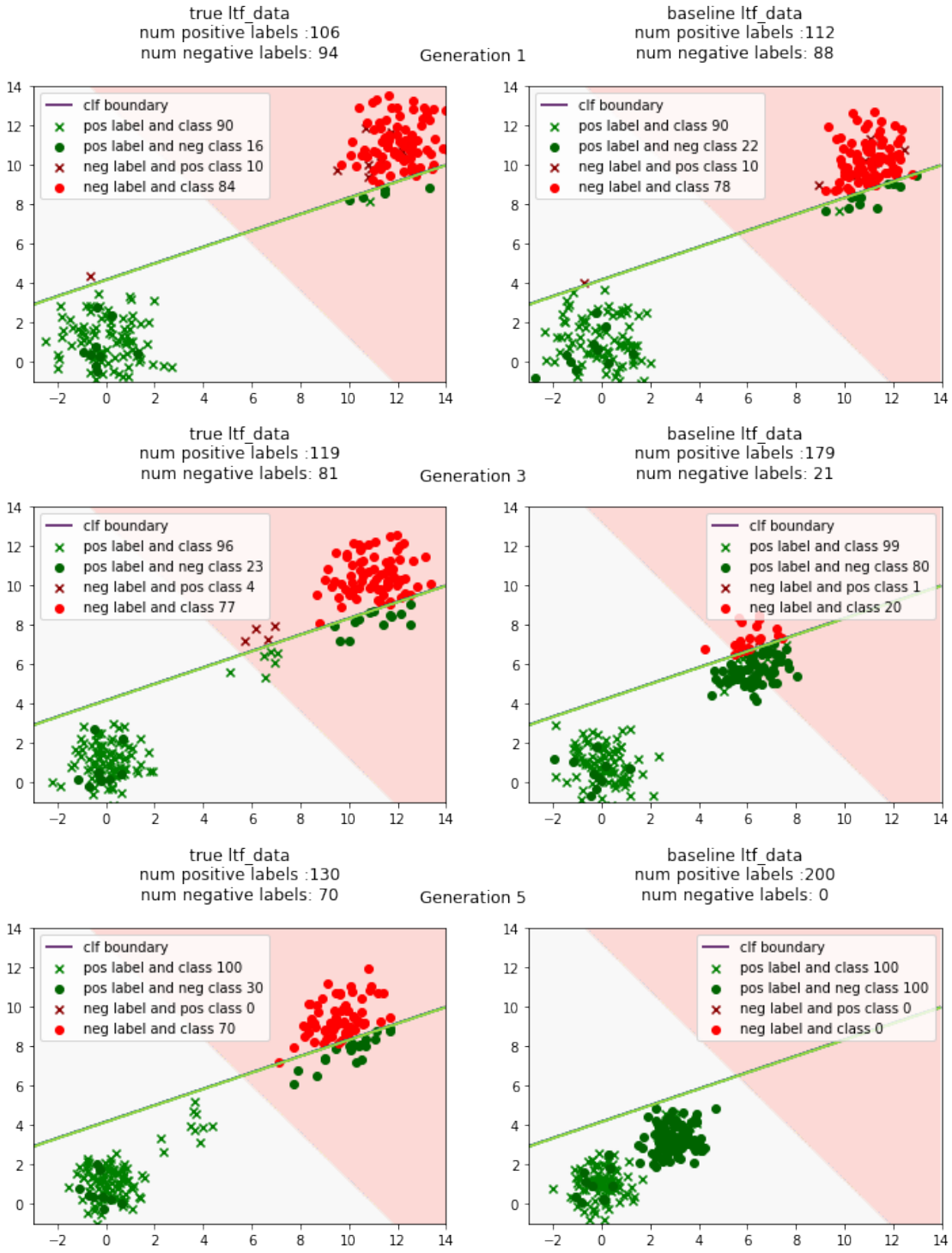

Figure 9: For the constrained decision function on group-wise generated data results are similar as for the unconstrained function. In both pipelines, all points converge towards the positive cluster. It is, however, much faster in the true data this time, because more points are labeled positive due to the constraint.