fairret: a Framework for Differentiable Fairness Regularization Terms

MaryBeth Defrance[1], Maarten Buyl[1] and Tijl De Bie[1]

[1]*Ghent University, Belgium*

Abstract

Current fairness toolkits in machine learning only admit a limited range of fairness definitions and have seen little integration with automatic differentiation libraries, despite the central role these libraries play in modern machine learning pipelines. We present a framework of fairness regularization terms (FAIRRETs) which quantify bias as modular, flexible objectives that are easily integrated in automatic differentiation pipelines. By employing a general definition of fairness through linear-fractional statistics, many group fairness definitions can be enforced. Experiments show minimal loss of predictive power compared to baselines. Our contribution includes a PyTorch implementation of the FAIRRET library.

Keywords

fairness, machine learning, library, automatic differentiation, fairness definitions

1. Introduction

The field of AI fairness has been concerned with formalizing ethical concepts of discrimination and bias in technical definitions that can be assessed and pursued in AI systems [1]. A popular paradigm for this formalization in binary classification is to use *group fairness* definitions [2], which require the model's predictions to treat people from different sensitive groups similarly.

Despite ample research on group fairness definitions and methods to achieve them, an easy-to-use and flexible implementation has not yet been realized. Popular fairness toolkits such as Fairlearn [3] and AIF360 [4] expect the underlying model in the form of *scikit-learn Estimators* [5] that can be retrained at-will in fairness meta-algorithms, but this aligns poorly with the paradigm of automatic differentiation libraries like PyTorch [6], which have become the bedrock of modern machine learning pipelines. These toolkits only integrate with automatic differentiation in their implementations of adversarial fairness [7], but these still require full control over the training process and lack generality in the fairness notions they can enforce.

We formally propose the FAIRRET framework in an effort to resolve these issues. At its core, the framework uses *fairness regularization terms* (FAIRRETs) that can be easily integrated into PyTorch-based pipelines (an example is given in Appendix A). They pursue any fairness definition expressed as a parity between statistics in a linear-fractional notation, which covers all group fairness definitions considered by Verma and Rubin [2]. Thus, all these definitions are fully compatible with any FAIRRET in any differentiable model.

EWAF'24: European Workshop on Algorithmic Fairness, July 01–03, 2024, Mainz, Germany
marybeth.defrance@ugent.be (M. Defrance); maarten.buyl@ugent.be (M. Buyl); tijl.debie@ugent.be (T. D. Bie)
0000-0002-6570-8857 (M. Defrance); 0000-0002-5434-2386 (M. Buyl); 0000-0002-2692-7504 (T. D. Bie)

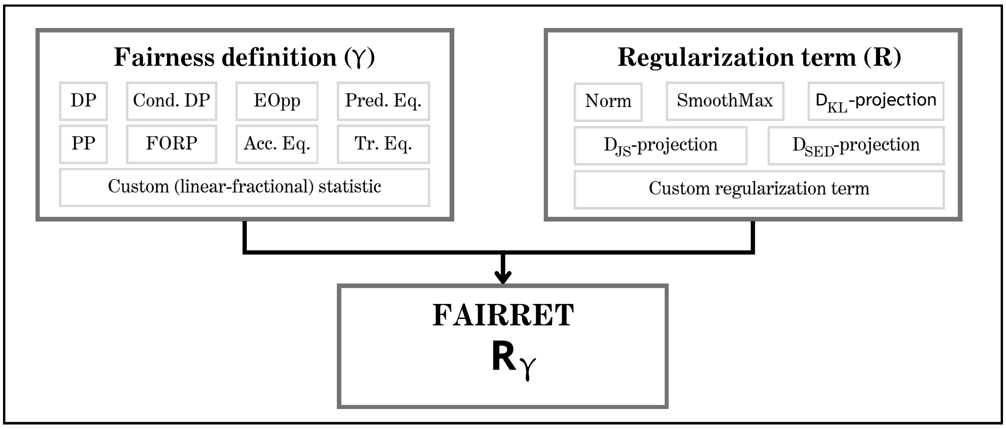

Figure 1: This diagram shows the modular nature of FAIRRET and provides an overview of the fairness definitions and methods present in the framework and notes the flexibility to implement novel ones.

In contrast to Fairlearn and AIF360, our proposed FAIRRETs act as a loss term that can simply be added *within* a training step. Two PyTorch-specific projects with similar goals as our paper are FairTorch [8] and the Fair Fairness Benchmark (FFB) [9]. However, neither present a formal framework and both only support a limited range of fairness definitions.

This work is an extended abstract of a full paper [10] presented at the *International Conference on Learning Representations* (ICLR) 2024. The implementation of our framework is available at https://github.com/aida-ugent/fairret, which we are currently extending into a full library.

2. How to build your FAIRRET

A FAIRRET is defined by two elements. First is the fairness definition it aims to satisfy. Second is the method used to evaluate the model with regard to that fairness definition. Figure 1 illustrates this combination and lists the definitions and methods already integrated into the framework.

2.1. Fairness definitions

Let $\mathbf{X} \in \mathbb{R}^{d_x}$ denote the feature vector of an individual, $\mathbf{S} \in \mathbb{R}^{d_s}$ their *sensitive* feature vector and $Y \in \{0, 1\}$ a binary output label. We want to learn a probabilistic classifier f such that its predictions $f(\mathbf{X})$ match Y while minimizing disparities over different $\mathbf{S}$. Our definition of sensitive features $\mathbf{S}$ as real-valued, d_s-dimensional vectors allows us to take a mix of multiple sensitive traits into account, both discrete and continuous. Categorical sensitive features are one-hot encoded, e.g. by encoding 'white' or 'non-white' as the vectors $\mathbf{S} = (1, 0)^\top$ and $\mathbf{S} = (0, 1)^\top$ respectively. The variable S_k denotes the kth sensitive feature.

We use a simplified version of the solution from Celis et al. [11] to translate fairness definitions as a parity between linear-fractional statistics γ:

$$\gamma(k; f) = \frac{\mathbb{E}[S_k(\alpha_0(\mathbf{X}, Y) + f(\mathbf{X})\beta_0(\mathbf{X}, Y))]}{\mathbb{E}[S_k(\alpha_1(\mathbf{X}, Y) + f(\mathbf{X})\beta_1(\mathbf{X}, Y))]} \tag{1}$$

Table 1
Fairness definitions and their α and β functions. Conditional Demographic Parity encompasses many notions with an arbitrary function ζ conditioned on the input $\mathbf{X}$.

Fairness Definition	α_0	β_0	α_1	β_1
Demographic Parity [12]	0	1	1	0
Conditional Demographic Parity [13]	0	$\zeta(\mathbf{X})$	$\zeta(\mathbf{X})$	0
Equal Opportunity [14]	0	Y	Y	0
False Positive Parity [14]	0	1 - Y	1 - Y	0
Predictive Parity [15]	0	Y	0	1
False Omission Parity	Y	-Y	1	-1
Accuracy Equality [16]	1 - Y	2Y - 1	1	0
Treatment Equality [16]	Y	-Y	0	1 - Y

with α_0, α_1, β_0, and β_1 functions that do not depend on $\mathbf{S}$ or f. Table 1 shows the statistic γ for a range of fairness definitions, defined through their α and β functions.

The set $\mathcal{F}_\gamma$ of probabilistic classifiers f that adhere to the fairness definition is expressed as

$$\mathcal{F}_\gamma \triangleq \{f : \mathbb{R}^{d_x} \to \{0,1\} \mid \forall k \in [d_s] : \gamma(k; f) = \bar{\gamma}(f)\}. \tag{2}$$

In other words, the statistic $\gamma(k; f)$ for each sensitive attribute S_k should equal the overall statistic $\bar{\gamma}(f) \triangleq \frac{\mathbb{E}[\alpha_0(\mathbf{X},Y)+f(\mathbf{X})\beta_0(\mathbf{X},Y)]}{\mathbb{E}[\alpha_1(\mathbf{X},Y)+f(\mathbf{X})\beta_1(\mathbf{X},Y)]}$ computed independently of the sensitive attributes. By fixing $\bar{\gamma}$ to a constant $c \in \mathbb{R}$, any fairness definition can be enforced with a **linear** constraint:

$$\gamma(k; f) = c \iff \mathbb{E}[S_k(\alpha_0(\mathbf{X}, Y) - c\alpha_1(\mathbf{X}, Y) + f(\mathbf{X})(\beta_0(\mathbf{X}, Y) - c\beta_1(\mathbf{X}, Y)))] = 0 \tag{3}$$

2.2. Regularization terms

The bias of a parameterized, probabilistic classifier h is quantified as a FAIRRET that can be minimized through automatic differentiation, in addition to any existing loss function $\mathcal{L}_Y$:

$$\min_h \mathcal{L}_Y(h) + \lambda R_\gamma(h) \tag{4}$$

where $R_\gamma(h)$ is the FAIRRET for the fairness definition with statistic γ and strength $\lambda \in \mathbb{R}_{>0}$.

The FAIRRET framework admits many kinds of regularizers, due to the practical form of the statistics γ. Two types are currently integrated, namely *violation* and *projection* FAIRRETS.

We first discuss the ***Norm*** FAIRRET, a type of violation FAIRRET:

$$R_\gamma(h) \triangleq \left\| \frac{\gamma(k; h)}{\bar{\gamma}(h)} - 1 \right\| \tag{5}$$

with $\|\cdot\|$ a norm over $\mathbb{R}^{d_s}$. Such a regularization term has been proposed several times [17, 18, 19], though without the same degree of modularity with respect to γ.

Second, an example of a projection FAIRRET is the D_{KL}**-projection**:

$$R_\gamma(h) \triangleq \min_{f \in \mathcal{F}_{\gamma(\bar{\gamma}(h))}} \mathbb{E}[D_{KL}(f(\mathbf{X}) || h(\mathbf{X}))] \tag{6}$$

with D_{KL} the *Kullback-Leibler* divergence. The FAIRRET maps h onto the *closest* fair model $f \in \mathcal{F}_{\gamma(\bar{\gamma}(h))}$. Projection FAIRRETS generalize some prior work [20, 21, 22] to all definitions with linear-fractional statistics, as they are enforced with linear constraints using Eq. (3).

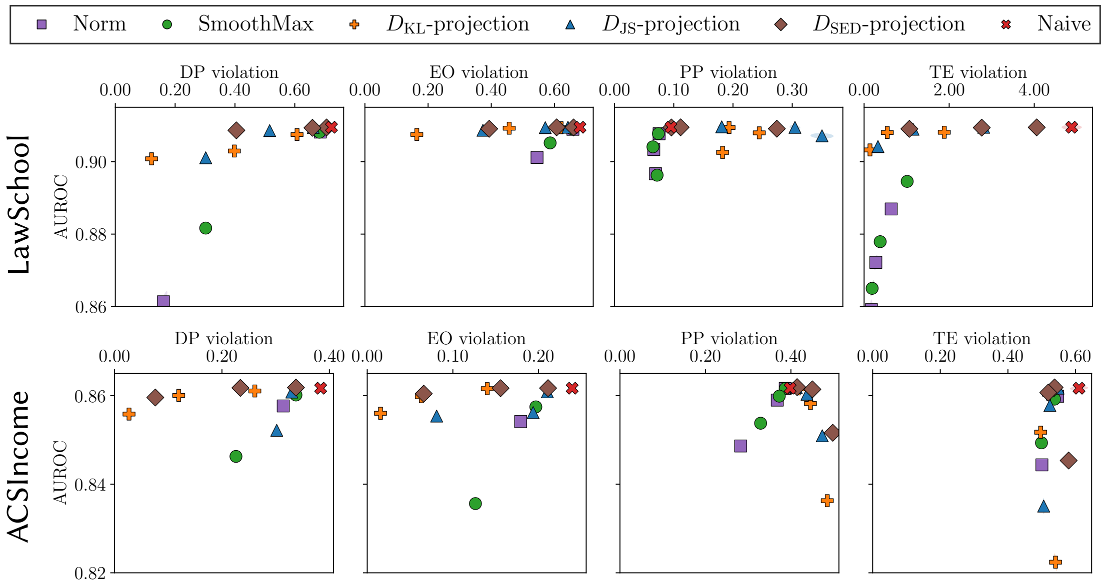

Figure 2: Mean test set results with confidence ellipse for the standard error (see Appendix B). Each marker is a separate combination of dataset, FAIRRET, FAIRRET strength, and statistic. Results in the top left are optimal. Failed runs (with an AUROC far worse than the rest) are omitted.

3. Experiments

Experiments were conducted on the *LawSchool*[1], and *ACSIncome* [23] datasets. Each dataset has multiple sensitive features, including some continuous. Figure 2 shows the results for the experiments. Each point represents a specific FAIRRET optimized for that statistic with a certain strength λ. An *Naive* baseline with $\lambda = 0$ is also included. In the full paper, the evaluation is done on two additional datasets and the FAIRRETS are compared to existing methods [10].

The results in Figure 2 show that the performance of a FAIRRET is dependent on the dataset itself and the fairness definition it aims to satisfy. The non-linear (yet still linear-fractional) fairness statistics like predictive parity and treatment equality seem more difficult to minimize. This leads us to conclude that not one FAIRRET can be chosen as the optimal solution, but rather that the best FAIRRET is dependent on the fairness definition and the dataset.

4. Conclusion

The FAIRRET framework allows for a wide range of fairness definitions by comparing linear-fractional statistics for each sensitive feature. We implement several FAIRRETS and show how they are easily integrated in existing machine learning pipelines utilizing automatic differentiation. More details can be found in the full paper [10].

[1]Curated and published by the SEAPHE project

Acknowledgments

The research leading to these results has received funding from the Special Research Fund (BOF) of Ghent University (BOF20/DOC/144 and BOF20/IBF/117), from the Flemish Government under the "Onderzoeksprogramma Artificïele Intelligentie (AI) Vlaanderen" programme, and from the FWO (project no. G0F9816N, 3G042220, G073924N).

References

[1] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A Survey on Bias and Fairness in Machine Learning, ACM Computing Surveys 54 (2021) 115:1–115:35. doi:10.1145/3457607.

[2] S. Verma, J. Rubin, Fairness definitions explained, in: Proceedings of the International Workshop on Software Fairness, ACM, Gothenburg Sweden, 2018, pp. 1–7. doi:10.1145/3194770.3194776.

[3] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, K. Walker, Fairlearn: A toolkit for assessing and improving fairness in AI, Technical Report MSR-TR-2020-32, Microsoft, 2020. URL: https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/.

[4] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, Y. Zhang, AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias, 2018. URL: https://arxiv.org/abs/1810.01943.

[5] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019.

[7] B. H. Zhang, B. Lemoine, M. Mitchell, Mitigating Unwanted Biases with Adversarial Learning, in: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 335–340. doi:10.1145/3278721.3278779.

[8] S. Masashi, Fairtorch, https://github.com/wbawakate/fairtorch, 2020. Version 0.1.2.

[9] X. Han, J. Chi, Y. Chen, Q. Wang, H. Zhao, N. Zou, X. Hu, FFB: A Fair Fairness Benchmark for In-Processing Group Fairness Methods, 2023. doi:10.48550/arXiv.2306.09468. arXiv:2306.09468.

[10] M. Buyl, M. Defrance, T. D. Bie, fairret: a framework for differentiable fairness regulariza-

tion terms, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=NnyD0Rjx2B.

[11] L. E. Celis, L. Huang, V. Keswani, N. K. Vishnoi, Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, ACM, Atlanta GA USA, 2019, pp. 319–328. doi:10.1145/3287560.3287586.

[12] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, Fairness through awareness, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, Association for Computing Machinery, New York, NY, USA, 2012, pp. 214–226. doi:10.1145/2090236.2090255.

[13] S. Wachter, B. Mittelstadt, C. Russell, Why fairness cannot be automated: Bridging the gap between eu non-discrimination law and ai, Computer Law and Security Review 41 (2020). URL: https://papers.ssrn.com/abstract=3547922. doi:10.2139/ssrn.3547922.

[14] M. Hardt, E. Price, E. Price, N. Srebro, Equality of Opportunity in Supervised Learning, in: Advances in Neural Information Processing Systems, volume 29, Curran Associates, Inc., 2016.

[15] A. Chouldechova, Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments, Big Data 5 (2017) 153–163. doi:10.1089/big.2016.0047.

[16] R. Berk, H. Heidari, S. Jabbari, M. Kearns, A. Roth, Fairness in criminal justice risk assessments: The state of the art, Sociological Methods & Research 50 (2021) 3–44. URL: https://doi.org/10.1177/0049124118782533. doi:10.1177/0049124118782533. arXiv:https://doi.org/10.1177/0049124118782533.

[17] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, C. Dwork, Learning Fair Representations, in: Proceedings of the 30th International Conference on Machine Learning, PMLR, 2013, pp. 325–333.

[18] M. Padala, S. Gujar, FNNC: Achieving fairness through neural networks, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI'20, Yokohama, Yokohama, Japan, 2021, pp. 2277–2283.

[19] M. Wick, s. panda, J.-B. Tristan, Unlocking Fairness: A Trade-off Revisited, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019.

[20] W. Alghamdi, S. Asoodeh, H. Wang, F. P. Calmon, D. Wei, K. N. Ramamurthy, Model Projection: Theory and Applications to Fair Machine Learning, in: 2020 IEEE International Symposium on Information Theory (ISIT), IEEE, 2020, pp. 2711–2716. doi:10.1109/ISIT44484.2020.9173988.

[21] D. Wei, K. N. Ramamurthy, F. Calmon, Optimized Score Transformation for Fair Classification, in: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 1673–1683.

[22] M. Buyl, T. De Bie, The KL-Divergence Between a Graph Model and its Fair I-Projection as a Fairness Regularizer, in: Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, 2021, pp. 351–366.

[23] F. Ding, M. Hardt, J. Miller, L. Schmidt, Retiring adult: New datasets for fair machine learning, Advances in Neural Information Processing Systems 34 (2021).

[24] P. Schubert, M. Kirchner, Ellipse area calculations and their applicability in posturography, Gait & Posture 39 (2014) 518–522. doi:10.1016/j.gaitpost.2013.09.001.

A. Code Use Examples

```
import torch
import torch.nn.functional as F
from fairret.statistic import TruePositiveRate
from fairret.loss.violation import NormLoss

# The TruePositiveRate class is a subclass of LinearFractionalStatistic.
statistic = TruePositiveRate()

# The fairret modules accept any LinearFractionalStatistic instance.
fairret = NormLoss(statistic)
fairret_strength = 1.0

def train_epoch(train_loader, model, optimizer):
    for feat, sens, target in train_loader:
        optimizer.zero_grad()

        logit = model(feat)
        bce_loss = F.binary_cross_entropy_with_logits(logit, target)
        fairret_loss = fairret(logit, feat, sens, target)
        loss = bce_loss + fairret_strength * fairret_loss
        loss.backward()

        optimizer.step()
```

Listing 1: Example use of the FAIRRET library in a simple PyTorch setup.

Listing 1 displays a code example of how the FAIRRET can easily be deployed in a typical PyTorch [6] setup. It suffices to simply load a subclass of `LinearFractionalStatistic` and pass it on to a FAIRRET implementation instance such as `NormLoss` (as defined in Def. 7). The FAIRRET is then used to compute the quantification of unfairness as a loss like any other in PyTorch. In this case, we use the true positive rate statistic to pursue the fairness notion of equalized opportunity (EO).

B. Confidence Ellipses

The confidence ellipses we use in Fig. 2 are uncommon in machine learning literature. Yet, they work well for our purpose of comparing trade-offs between metrics that may be noisy depending on randomness during training and dataset split selection.

Recall that 1-dimensional confidence intervals typically assume a mean estimator to be normally distributed. The confidence interval then denotes the uncertainty of the sample mean using the standard error. Similarly, confidence ellipses assume a 2-dimensional point, i.e. the 2-dimensional mean estimator, to have a multivariate normal distribution that can be characterized through the sample mean and standard error statistics.

Our implementation of the confidence ellipses follows a featured implementation on `matplotlib`[2]. However, a crucial difference is that this implementation computes a confidence interval for a 2-dimensional random variable based on the covariance matrix for the standard *deviation* of

[2]https://matplotlib.org/3.7.0/gallery/statistics/confidence_ellipse.html.

samples of that variable. Following observations by Schubert and Kirchner [24], we instead want to show the uncertainty of the mean estimator, which should use the standard deviation of that estimator, i.e. the covariance for the standard *error*. This is accomplished by dividing the covariance matrix in the `matplotlib` implementation by the number of seeds (5) we use in our experiments.