Notes on risk

April 12, 2024

Contents

1 Overview

The BRIO system features a module devoted to the measurement of the risk related to fairness violations by AI systems. The risk measure produced aggregates the results of all available relevant tests detecting fairness violations relative to an AI system. The risk module, therefore, takes in input a series of n different test results, relative to possibly different sensitive features, and returns one value in the real unit interval $[0, 1]$ which represents how high in the interval $[0, 1]$ is the risk that the tested AI system behaves in an unfair manner.

As the BRIO bias detection module does not only compare the behaviour of the AI algorithm on the classes relative to the sensitive feature but can also execute similar checks on possibly several subclasses, in general the result of one fairness test will consist of m lines. Each one of these lines will be relative to a subclass of the considered classes. Suppose, for example, that our sensitive feature is `gender`, the detection module of BRIO will not only compare the behaviour of the AI algorithm on the classes obtained by selecting a particular value of `gender`, but will also compare the behaviour of the AI algorithm on the subclasses obtained by fixing the values of features different from `gender` and by varying the value of `gender`. For instance, one line of the output will be about the behaviour of the AI system on the class of male individuals as compared to its behaviour on the class of female individuals, another line will be about the behaviour of the AI system on the subclass of rich male individuals as compared to its behaviour on the class of rich female individuals, yet another line will be

about the behaviour on the subclass of poor male individuals as compared to its behaviour on the class of poor female individuals, and so on.

Therefore, each line of the output will provide the following information:

- the set of non-sensitive feature values used to determine the considered subclasses, if any,
- the number of elements of the union of all considered (sub)-classes,
- the value of the divergence for the considered (sub)-classes,
- the threshold employed.

This information will be used to compute the risk measure emerging from a series of tests. In computing this measure, it is also possible to choose whether to focus on group fairness or individual fairness. Intuitively, focusing on *group fairness* means deeming more serious a discrimination based on very little information. For instance, a choice made only on the basis of the value of the sensitive feature will be a group discrimination. Focusing on *individual fairness*, on the other hand, means deeming more serious a discrimination between two individuals which have many values in common but a different value relatively to the sensitive feature.

1.1 Risk associated to a single test

If n tests are available, the risk coefficient for a given test $i \in \{1, \ldots, n\}$ is computed as follows,

$$\mathrm{R}_i = \sum_{j=1}^{m} \delta(i,j) \cdot q(i,j) \cdot \sqrt[3]{|e(i,j)|} \cdot \sqrt[3]{\varepsilon(i,j)} \cdot w(i,j)$$

where m is the number of lines in the output of test i and

1. $\delta(i,j) = 1$ if line j of test i is about a violation of fairness, and $\delta(i,j) = 0$ otherwise,
2. $q(i,j)$ is the number of elements in the union of the two classes (or subclasses) used for the comparison relative to line j over the total number of datapoints,
3. $e(i,j)$ is the distance between the divergence and the threshold relative to line j,
4. $\varepsilon(i,j)$ is the threshold employed at line j,
5. $w(i,j)$ is the weight of the possible fairness violation relative to line j.

Hence, $\delta(i,j)$ simply sets the addend relative to line j to 0 if line j does not correspond to a fairness violation, $q(i,j)$ makes the addend proportional to the number of individuals involved in the possible violation over all individuals, $e(i,j)$ makes the addend proportional to the gravity of the violation in terms of distance from the threshold, $\varepsilon(i,j)$ makes the addend inversely proportional to the strictness of the threshold employed. Factors involving e and ε are typically[1] two or three orders of magnitude smaller than the others, so we scale their weight taking their cube root.

The weight $w(i,j)$ of the violation depends, in turn, on whether one focuses on *group fairness* or on *individual fairness*. In the first case, the weight increases if the possible fairness violation concerns a class determined by a few features (thus, a rather general class). In the second case, the weight increases if the possible fairness violation concerns a class determined by many features (thus, a rather specific class).

1.2 Risk associated to a battery of tests

The risk measurement function can be formally defined as follows:

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \mathrm{R}_i$$

where n is the number of executed tests and R_i is as above.

2 Execution of the module

The risk module can be executed in two modes,

1. the *standard* mode: the user is only required to provide which features in the database are deemed 'sensitive',

2. the *customizable* mode: the user is required to provide the sensitive features, the tolerance (`high`/`low`), the choice to favour either `group` or `individual` fairness, the aggregating function for featured with multiple classes (`max`/`average`/`min`).

The standard mode is simply an execution of the customizable mode with fixed parameters: `high` and `average`, and a message calculating the average and `group` and `individual`, with it broken down to each.

2.1 Pipeline

Assume then that all parameters in the section above are defined. We only describe the process of checking *one* sensitive feature `f`: when more are required,

[1] Using the automated threshold they are, but it is always possible to use customized thresholds. Notice that the closer that is to 1, the smaller is the effect of taking the cube root.

the user gets in output a list of each individual risks and their minimum, maximum, and average.

Moreover, assume that we have

- a fixed upper bound for the execution time of single checks (=test line computations, cf. 2.2), and call this `T`,
- a fixed lower bound for meaningful correlation indexes, and call this $\mathtt{C}_0$.

.

Then the risk module

1. computes the correlation between the features and the output, and orders the features in order of decreasing correlation;
2. selects only features that show correlation above $\mathtt{C}_0$;
3. computes the greatest maximum runtime (cf. 2.2) of the algorithm with sensitive feature `f` and conditioning those selected in 2;
4. returns a message showing the number of conditioning features and the upper bound in 3, and asks the user if they want to proceed;
5. if the user wants to proceed, it executes the module and returns the risk value, if not, it asks the user to select a new $\mathtt{C} > \mathtt{C}_0$ and returns to 2.

2.2 Strategy

Now we only need to provide a suitable way of using the `freqvfreq` option in our bias module. The check proceeds breadth first. For a line in the output (that is, the js in 1.1) the algorithm

1. starts computing the line,
2. if it succeeds in time $\leq \mathtt{T}$, it moves on to 4, if not
3. it records the 'bad' combination of features, and closes it for extension[2],
4. $j = j + 1$,
5. if the combination in j is in the 'bad' record, go straight to 4,
6. go back to 1.

This way, it is possible to combinatorially estimate an upper bound for the computation time[3] (possible combinations $\times$ `T`), and depending on the user's computational capability they can decide whether they want to proceed to the calculus or not. This procedure also guarantees that failure of a check aborts all possible checks that are 'extensions' of it, so that, all in all, the runtime will usually be *way below* that calculated.

[2]Meaning that if we have conditioning $\{a, b, c, d, e\}$ and it fails at abc, then to the 'bad' record one adds $\{abc, abcd, abce\}$, then iterate...

[3]Modulo everything else, that should not account for much.

References

[CDG+23] Greta Coraglia, Fabio Aurelio D'Asaro, Francesco Antonio Genco, Davide Giannuzzi, Davide Posillipo, Giuseppe Primiero, and Christian Quaggio. Brioxalkemy: a bias detecting tool. In Guido Boella, Fabio Aurelio D'Asaro, Abeer Dyoub, Laura Gorrieri, Francesca A. Lisi, Chiara Manganini, and Giuseppe Primiero, editors, *Proceedings of the 2nd Workshop on Bias, Ethical AI, Explainability and the role of Logic and Logic Programming co-located with the 22nd International Conference of the Italian Association for Artificial Intelligence (AI*IA 2023), Rome, Italy, November 6, 2023*, volume 3615 of *CEUR Workshop Proceedings*, pages 44–60. CEUR-WS.org, 2023.

[KL51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[Lin91] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.