How to Make Machine Learning Financial Recommendations More Fair: Theoretical Explanation of Empirical Results

Tho M. Nguyen, Saeid Tizpaz-Niari, and Vladik Kreinovich

Abstract Machine learning has been actively and successfully used to make financial decisions. In general, these systems work reasonably well. However, in some cases, these systems show unexpected bias towards minority groups – the bias that is sometime much larger than the bias in the data on which they were trained. A recent paper analyzed whether a proper selection of hyperparameters can decrease this bias. It turned out that while the selection of hyperparameters indeed affect the system's fairness, only a few of the hyperparameters lead to consistent improvement of fairness: the number of features used for training and the number of training iterations. In this paper, we provide a theoretical explanation for these empirical results.

1 Formulation of the General Problem

Machine learning have become ubiquitous in decision making. In many practical situations, we need to make a recommendation. For example, if a person applies for a loan, the bank needs to decide whether providing this loan is worth a risk, and if yes, shall the bank give the full requested amount or a smaller amount. Similar decisions need to be made in other financial situations: e.g.

- when a start-up company applies for funding, or
- when a well-established company wants the bank to finance a project.

Tho M. Nguyen
Faculty of Banking and Finance, Ho Chi Minh City Open University, Ho Chi Minh City, Vietnam, e-mail: tho.nm@ou.edu.vn

Saeid Tizpaz-Niari and Vladik Kreinovich
Department of Computer Science,University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA, e-mail: saeid@utep.edu, vladik@utep.edu

In all such situations, there is no ready formulas for making a decision. So, we have to rely on specialists. Some specialists are more skilled, some are less skilled. It would therefore be nice to incorporate the expertise of skilled experts into a computer system – this would help less experienced decision makers make good decisions.

In some cases, experienced experts can formulate their decision making process in terms of precise rules, but in financial domain, such situations are rare. In many cases, the decisions of experienced experts are motivated largely by their intuition, intuition that they cannot describe in precise terms.

Since we do not have the rules, all we know about the decision making of experienced experts is a set of cases in which they made decision, i.e., the set of pairs (x_k, y_k), $k = 1, 2, \ldots$, where:

- x_k is the information the decision makers have about the k-th case, and
- y_k is the resulting decision.

Based on these pairs, we need to come up with an algorithm $f(x)$ that would, given a general input x, produce a reasonable decision $y = f(x)$. For the cases in which experienced decision makers made reasonable decisions, this algorithm should generate the same (or practically the same) decisions, i.e., we should have $f(x_k) \approx y_k$ for all k.

In computer science, the problem of generating an algorithm $f(x)$ based on several known examples (x_k, y_k) is known as *machine learning*. In statistics, this problem is also known as *regression*.

In the last decades, many effective machine learning techniques have been developed [1, 2], including deep learning. These techniques are actively – and successfully – used in many application areas, including finances. Banks and other financial institutions are using systems based on machine learning to decide whether to give a loan to a person.

Comment. Such systems are used in many other applications – e.g., such systems are used to decide whether a person who has served a significant portion of his/her sentence should be given a parole or this person needs more time to reform. In this paper, we concentrate on the financial applications, but the same ideas and results can be (and are) used in other applications as well.

Main problem: insufficient fairness. In many cases, the current software packages produce reasonable results. However, many of these packages have a problem with fairness:

- while they produce good results on average,
- for applicants from some minority groups their results are often not fair at all.

There have been many cases when the system were tested, and it turned out that the chances of a person to get a loan significantly increases if his/her ethnicity and/or race was changed in the application – or if we replace a female name with a male name; see, e.g., [4] and references therein.

How can we make the systems more fair?

2 How Can We Make the Corresponding Systems More Fair: A General Idea and What Happened When This Idea Was Tested

Is there an easy solution? At first glance, it may seem that with machine learning approach, there is not much that we can do other than changing the general machine learning algorithms. Algorithms for training a machine learning system (such as a neural network) are themselves not biased. Maybe the inputs are somewhat biased, but this does not explain the bias in the machine learning results: this bias is often much larger than the bias in the samples on which these systems have been trained.

So, at first glance, the problem is complex: we need to replace the current machine learning training algorithms with new algorithms:

- that would not simply generalize the available data, but
- that would also explicitly take into account the need for fairness.

This is a challenging research task, and so far, researchers have not produced such new algorithms.

But there is hope. The above – somewhat pessimistic – view comes from the viewpoint of an outsider, to whom machine learning (especially deep learning) is a magic tool: you give it data and it immediately generates great results.

- Such magic happens sometimes.
- However, as researchers who actually apply machine learning know very well, such ideal situations are rare.

The truth is that each machine learning tool comes with many parameters that we need to select to start training. These parameters are called *hyperparameters*, to distinguish them from the parameters describing the resulting model $y = f(x)$.

For example, to train a neural network, we need to decide how many layers to use, how many neurons to have in each layer, when to stop training, etc.

- For some values of these hyperparameters, we get spectacular results.
- However, for some other values, the system does not learn at all.

Since the proper choice of hyperparameters significantly affects the training result, why not try to see if a proper choice of hyperparameters would make the resulting systems more fair?

This idea was tested. An extensive research is this direction was actually performed [4], and here is a brief description of the results:

For most of hyperparameters, there seems to be no clear direction of the resulting effect. Change may be drastic, but in some cases, they go in one direction, and sometimes they go in an opposite direction. For example, sometimes placing the neurons in fewer layers makes the trained system significantly more fair, while in other case, a similar transformation further decreases the original system's level of fairness.

There are only a few hyperparameters that consistently affect fairness the same way. In a nutshell, these hyperparameters are related:

- to the number of features that is taken into account, and
- to the number of training iterations.

In both cases, the larger these hyperparameters, the more fair the resulting trained system.

Remaining problem. How can we explain these empirical results? This is what we do in this paper.

3 How We Explain the Empirical Results

Explaining the effect of number of features is straightforward. Of course, the more features we take into account, the more information we get about a person, the smaller is the effect of each individual feature.

Crudely speaking, if all the banking system know is the applicant's gender, then this will be the only information that it can use to make a decision on whether to give a loan or not. On the other hand, if the banking system takes 10 different features into account, gender being one of them, then we expect that only 10% of its decision will be affected by gender. And this is exactly what the above-cited empirical study observed: that the more features we take into account, the more fair the results.

But how can we explain the effect of the number of iterations? In contrast to the effect of the number of features, the effect of the number of iterations is somewhat puzzling. The more iterations we perform, the closer the values $f(x_k)$ generated by the trained model will be to the corresponding values y_k. In other words, as we increase the number of iterations, the closer the trained system will be to the original expert decisions. But why would the resulting trained system be more fair?

For some time, this was a puzzle for us, until we came up with a reasonable explanation.

Towards our explanation. From the informal mathematical viewpoint, what training means is minimizing the difference between the values $f(x_k)$ generated by the model and the corresponding values y_k.

- At first, before training, the values $f(x_k)$ are very different from the desired values y_k.
- As we train, the difference becomes smaller and smaller.

This is especially true for neural networks, in which each training step is, in effect, an application of gradient descent to minimize the objective function describing this difference.

How is this difference described? In the traditional neural networks, the difference d was described by the Least Squares expression (which is typically used in statistical regression; see, e.g., [3]):

$$d = \sum_k (f(x_k) - y_k)^2. \tag{1}$$

In modern machine learning algorithm, usually, somewhat more sophisticated expressions are used. However, the gist of our explanation does not depend on what expression we use. So, for simplicity, we will illustrate our idea on the example of the Least Squares expression (1).

Our explanation. We usually stop iterations when the value of the difference (1) becomes sufficiently small, i.e., when the value (1) does not exceed some threshold t.

How do we usually select this threshold? Suppose that we want to predict the desired values y_k with accuracy 10% (or, in case we taking about "yes"-"no" decision – with the probability of 90%). In other words, we want to have $|f(x_k) - y_k| \le 0.1 \cdot y_k$ for all k. This means that we want to have $(f(x_k) - y_k)^2 \le 0.01 \cdot y_k^2$ for all k. Thus, for the sum (1), we want to have

$$d = \sum_k (f(x_k) - y_k)^2 \le 0.01 \cdot \sum_k y_k^2.$$

So, in this case, it is reasonable to select a threshold $t = 0.01 \cdot \sum\limits_k y_k^2$.

Usually, all the values y_k have about the same order of magnitude $y_k \approx y$, so this natural threshold takes the form $t = 0.01 \cdot K \cdot y^2$, where by K, we denoted the overall number of training cases. So, after we apply this stopping criterion, we get approximate equality

$$d = \sum_k (f(x_k) - y_k)^2 \approx t = 0.01 \cdot K \cdot y^2. \tag{2}$$

Suppose now that we have a minority population that forms about 10% of the total. This means, in particular, that out of K cases, approximately 10% deal with applicants from this minority population. Let S denote the set of all indices k corresponding to this minority group, and let M denote the number of such indices (so that $N \approx 0.1 \cdot K$).

Based on the approximate equality (2), what can we conclude about the recommendations corresponding to folks from this minority population? The only thing that we can conclude is that the sum

$$\sum_{k \in S} (f(x_k) - y_k)^2$$

corresponding to applicants from this population does not exceed the overall sum, i.e., that

$$\sum_{k \in S} (f(x_k) - y_k)^2 \le 0.01 \cdot K \cdot y^2. \tag{3}$$

If we denote the average values of the difference $|f(x_k) - y_k|$ corresponding to minority population by δ, then (3) implies that $M \cdot \delta^2 \le 0.01 \cdot K \cdot y^2$. Since $M \approx 0.1 \cdot K$, this implies that $\delta^2 \le 0.1 \cdot y^2$ and thus (by taking the square root of both sides), that $\delta \le 0.3 \cdot y$.

So:

- while on average, the trained system reproduces the original decisions with accuracy 10%,
- for minority folks, the only thing we can guarantee is that the accuracy will be not larger than 30%.

And if the accuracy *is* 30% for all the minority folks, it can still lead to 10% accuracy in general.

What happens if we increase the number of iterations? In effect, this means that we further decrease the threshold. In the above example, if we decrease the threshold to $0.001 \cdot K \cdot y^2$, then the resulting inequality

$$\sum_{k \in S} (f(x_k) - y_k)^2 \le 0.001 \cdot K \cdot y^2$$

will guarantee the average-10% accuracy for the minority population as well – and will thus lead to more fair results.

So, an increase in the number of iterations indeed leads to more fairness.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, by a grant from the Hungarian National Research, Development and Innovation Office (NRDI), and by the Center of Excellence in Econometrics, Faculty of Economics, Chiang Mai University, Thailand.

The authors are greatly thankful to Hung T. Nguyen for his support and encouragement.

References

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.
3. D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.

4. S. Tizpaz-Niari, A. Kumar, G. Tan, and A. Trivedi,“Faireness-aware configuration of machine learning libraries”, *Proceedings of the 44th International Conference on Software Engineering ICSE’22, Pittsburgh,Pennsylvania, USA, May 21–29, 2022*, ACM Publ., New York, 2022.