Fair Universe Toy Problem Design

Mathis Reymond - mathis.reymond74@gmail.com

https://github.com/Mathisnplus1/fair-universe

Relevant links :

1. Main Notebook where we provide baselines to solve the problem : https://colab.research.google.com/drive/1jcZdpSRYY9K6L_pFiyx8NPdVG40aIluh
2. Didactic notebook where we show how to generate data : https://colab.research.google.com/drive/1JK4t2EmIdv5AgnyNlPD7iEBSHoUKkUFd#scrollTo=78c395ba

1 Background and motivation

Machine Learning has conquered many fields of science and has proven to be a very efficient tool to solve many interesting problems which in the past were considered impossible. Likewise High-Energy Physics has adopted Machine Learning [2]. Since 1990, Machine Learning is being used for particle identification [6]. A Machine Learning competition (HiggsML) to identify Higgs Boson [1] has recently been organized to use Machine Learning techniques to classify signal *vs.* background events.

In particle physics, datasets are generated using simulators. There are systematic uncertainties, which introduce bias in the datasets. It is of great importance to design Machine Learning algorithms, which take into account the bias and do uncertainty-aware classification of signal *vs.* background events [3]. Our goal is to create simple toy datasets with different nuissance parameters and create a machine learning competition on top of them, for uncertainty-aware classification of events. This competition will be hosted on the Codabench an open source platform to organize machine learning challenges [8]. Later on, this work will provide basis for developing international challenges and benchmarks in the context of the **"Fair Universe"** project, with similar purpose and goal.

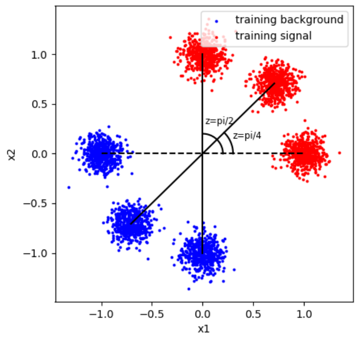

Figure 1: Randomly generated 500 signal and background points for three distinct values of the nuisance parameter $z = 0, \frac{\pi}{4}, \frac{\pi}{2}$. In this particular case, the standard deviation is set to 0.1.

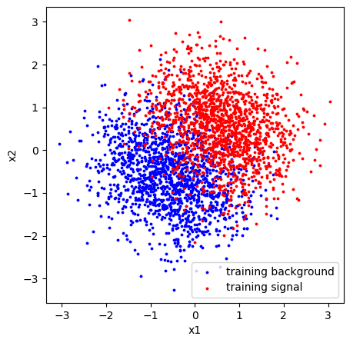

Figure 2: Randomly generated 500 signal and background points for three distinct values of the nuisance parameter $z = 0, \frac{\pi}{4}, \frac{\pi}{2}$. The standard deviation is set to 0.7

2 Data and task description

2.1 Data

For the proposed competition, we create our own data. The dataset consists of 2D probabilistic generated points (that is with feature spaces that are one or two dimensional). Each point/event belongs to one of the two classes : background ("b") and signal ("s"). These 2D points/events are represented by vectors of two features x_1 and x_2 corresponding to the coordinates x_1 and x_2 of the points. The points are generated by a continuous random distribution. The distribution is configurable and can be chosen to be Gaussian or exponential. The final distribution of the data will be decided based on similarity with data generated by ATLAS simulators. The dataset is structured as a Pandas DataFrame [7] with three columns "x_1", "x_2", and "y" for x_1-coordinate, x_2-coordinate, and target class respectively.

For initial experiments, we are generating 10,000 events for each class and in total 20,000 datapoints. In realistic High-Energy physics scenario, the number of background events are much larger than that of signal events, which makes the classification problem a lot more difficult.

2.2 Data in literature

Papers show other ways to introduce nuisance parameter. For instance, in [4], signal and background events are generated according to $2D$ Gaussian distributions. Background Gaussian is centered in $[0, 0]$ with covariance matrix

$\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$. The noise is introduced as a random continuous variable z following $\mathcal{N}(0,1)$. Then signal Gaussian is centered in $[1, 1+z]$ with the identity as covariance matrix.

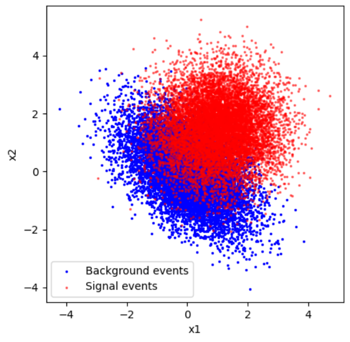

Figure 3: Scatter plot of 10000 background events and 10000 signal events as in [4] for $z = 0.5$

Even though this is a reasonable way to simulate systematic as founded in real life HEP scenarios, in first instance, we want our two features to be independent for the sake of simplicity and data generation suggested by [4] seems slightly too elaborated for our purpose.

2.3 Task

The task associated with the dataset is 2-class classification problem i.e. to classify if a point belongs to class background (b) or class signal (s). As the nuisance parameter z produced uncertainty, and additional task is to reduce the impact of paramater z as much as possible. In real life scenario these tasks are difficult because of two reasons: 1) the number of examples for signal and too small compared to background events; 2) Multiple nuisance parameters affect the data generation and the reduction of their impact is quite difficult.

2.4 Metric

Our model classifies events between signal and background and the output is two probabilities i.e. numbers between 0 and 1, one for signal and the other one for background. The final decision to predict a class i.e. class s or class b made based on a threshold. It is usually set to $\frac{1}{2}$, which means that if the signal output probability is larger than $\frac{1}{2}$, the event is classified as signal, else, it is classified as background. However, other values can be relevant according to the importance of false positive and false negative rates. The Receiver Operating Characteristic (ROC) curve is the plot of true positive rate over the false positive rate for different values of this threshold. We then compute Area under the ROC Curve which is our final metric. We use the implementation by scikit-learn [5]

3 Challenge Design

This section describes the design of Fair Universe challenge. Our challenge consists of 3 phases:

1. **Public**: participants are provided with a public dataset to experiment on their own machines
2. **Feedback**: participants needs to submit their code which is then evaluated on the challenge platform on a private dataset. Participants can submit 5 submissions per day with a total of 100 submissions.
3. **Final**: the last submission of participants is automatically submitted to the Final phase and evaluated on an unseen private dataset. Only one submission is allowed in this phase. Participants are evaluated and ranked based on the performance in this phase.

Our challenge design consists of a starting kit described below:

- **Sample Dataset**: a very small dataset for participants to get started
- **Getting Started Notebook**: a jupyter notebook to give an overview of the problem, how to load data, sample baselines, how to use ingestiona and scoring program, and how to submit code to CodaLab
- **Ingestion Program**: a python script which loads data and runs baseline model on the data to generate results/predictions
- **Scoring Program**: computes scores from the predictions made by ingestion program
- **Dataset 1**: a public dataset for participants to experiment withh
- **Dataset 2**: a private dataset to be used in the Feedback phase
- **Dataset 3**: a private dataset to be used in the Final phase

4 Nuisance

In this section we discuss the two kinds of nuisance we are able to introduce in the data. We also explain how to remove it as a preprocessing step, without knowing the class of the points.

4.1 x1-translation

The centers of distributions are on the x1-axis. The nuisance parameter appears as a translation, that is, each point $(x1, 0)$ is translated to $(x1+z, 0)$. To remove this bias from a set of points with both signal and background events, we can normalize the points. This consists in translating the points so their barycenter is at the origin and then scaling them so the standard deviation equals 1. See 7.

4.2 Rotation

This data generation process is inspired by [3]. We generate the points from a Gaussian distribution where the center of signal distribution is $(\cos(z), \sin(z))$ while the center of background distribution is $(-\cos(z), -\sin(z))$ for $z \in \left(0, \frac{\pi}{2}\right)$. The nuisance parameter z is the angle between the horizontal axis and the line going through the centers of the signal distribution and background distribution. We demonstrate the data generation in Figure 1 to better understand the effect of nuisance parameter z. In Figure 2 we have a scatter plot of signal and background points.

To remove this bias, we calculate the average absolute coordinates which have to be approximately equal to the coordinates of the center of signal distribution. Then we retrieve the nuisance parameter. Finally we apply a negative rotation of this angle to all the points.

5 Baselines results

5.1 Classifier used

5.1.1 Optimal Bayes Classifier

One of the biggest benefits with working in small dimension is the ability to compute analytically optimized solutions. Thus we start treating the problem in one dimension to analytically find the Optimal Bayes Classifier (OBC). That it to say, knowing the parameters of two continuous random distributions, it is possible to analytically compute the optimal classifier in the sens of Bayes. We consider two 1-dimensional continuous random distributions, one standing for background and the other one for signal. We assume that signal feature is constrained into the range $[m, M]$. Being given an event with feature x, we can calculate the probability p_s that it is signal and the probability p_b that it is background using Bayes formula :

$$p_s := p(y = 1|X = x) = \frac{p(y = 1)p(X = x|y = 1)}{p(X = x)} \tag{1}$$

$$p_b := p(y = 0|X = x) = \frac{p(y = 0)p(X = x|y = 0)}{p(X = x)} \tag{2}$$

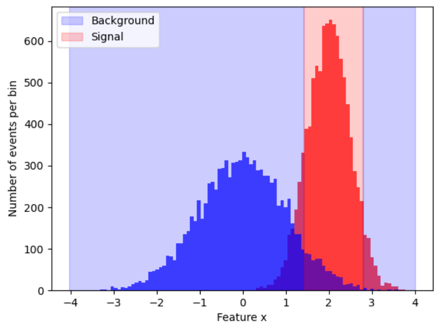

Figure 4: OBC regions estimated for randomly distributed 500 signal events with $\mathcal{N}(2, 0.5)$ and 500 background events with $\mathcal{N}(0, 1)$

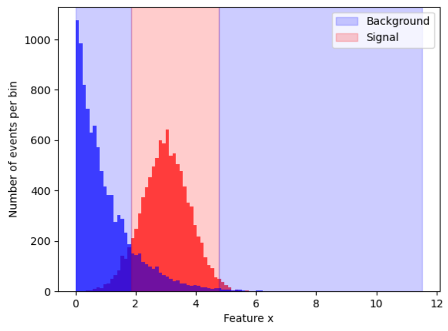

Figure 5: OBC regions estimated for randomly distributed 500 signal events with $\mathcal{N}(3, 0.8)$ and 500 background events with exponential distribution with scale factor set to 1

Then we rule the classifier so that it assigns signal class to the event if $p_s > p_b$ and background class otherwise.

Then, being given a set of (one-dimensional) signal events, that we assume to follow known random distribution, we derive its parameters using maximum likelihood estimation. We assume the same applies to background events. Knowing these approximated parameters we compute the OBC as above. See fig. 4, 5

We also implemented 2D optimal classifier 6.

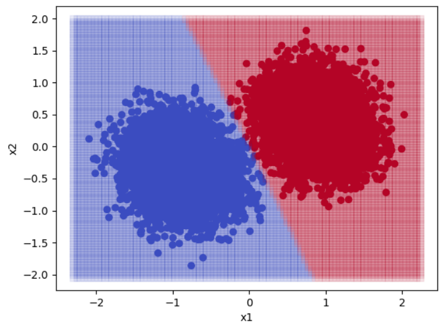

Figure 6: 2D OBC of events plagued with rotation bias

5.1.2 Gradient Boosted Classifier

Another classifier that we use in the following baselines is the Gradient Boosted Classifier (GBC) provided by scikit-learn library. We use its default parameters.

5.2 Baselines

Let's go back to the 2D problem. We implemented the 2D OBC. Here we introduce three approaches to tackle the problem. For each of them, we introduce the rotation nuisance and generate 10000 signal and 10000 background train events. These events will be generated in different ways depending on the baseline. We also generate 1000 signal and 1000 background test events that will be used to evaluate and compare the baselines. Both signal and background points are generated with Gaussian distribution with covariance matrix equal to $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

5.2.1 Ignoring nuisance

The very first attempt consist in ignoring the nuisance. We generate 10000 train events for the average nuisance value, that is $z = \frac{\pi}{4}$.

5.2.2 Data Augmentation

For this baseline, we generate 1000 train events for 10 nuisance values regularly chosen between 0 and $z = \frac{\pi}{2}$, that is $z_1 = 0, z_2 = \frac{\pi}{20}, ..., z_{10} = 9\frac{\pi}{20}$. Then we train the classifiers with these points. With this method, we expect the classifier to generalize better.

5.3 Removing bias as reprocessing step

For this baseline, we generate the points in the exact same way as the previous one. But instead of directly training the classifier, we first apply the rotation-debiasing-procedure.

5.3.1 Baselines comparison

Table 1: AUC scores for each baseline method with rotation nuisance

Baseline	**OBC's AUC Score**	**GBC's AUC Score**
Ignoring nuisance	0.75	0.88
Data Augmentation	0.74	0.88
Removing bias	0.77	0.90

We observe on 1 that the last baseline method performs slightly better than the two others. It is more clear if we introduce a translation nuisance instead of the rotation one. Indeed, if the signal Gaussian distributions are centered at $(4+z_i, 0)$ and the background ones at $(6+z_i, 0)$ where $z_1 = 0, z_2 = \frac{1}{5}, ..., z_{10} = \frac{9}{5}$, see 7, then we obtain the following AUC scores 2.

Table 2: AUC scores for each baseline method with translation nuisance

Baseline	**OBC's AUC Score**	**GBC's AUC Score**
Ignoring nuisance	0.71	0.87
Data Augmentation	0.70	0.88
Removing bias	0.82	0.93

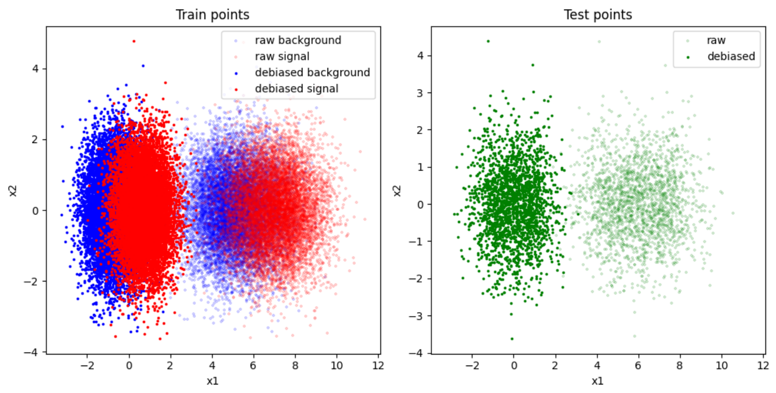

Figure 7: Train and test points and their corresponding debiased points for translation nuisance

6 Conclusions and future work

To conclude, we made a notebook that setups the 2D nuisance classification problem and provide baseline to solve this problem. We are able to generate the data for the challenge, although it still needs to be discussed whether contestants will be provided an explicit data set or tools to make their own.

Further exploration remain needed for more relevant ways to introduce nuisance parameter. Moreover, there are different directions which can be discovered to solve the problem e.g. treating the nuisance parameter z as an explicit input of the network along and treating it as a feature. Adversarial Learning is another direction where one classifier learns to classify data points while another network uses the output of the classifier to learn about the nuisance parameter. The goal is to minimize classification loss and maximize adversarial loss. In addition, designing a new metric to evaluate the bias treatment is also very important to be taken care of.

References

[1] Claire Adam-Bourdarios et al. “The Higgs boson machine learning challenge”. In: *Proceedings of the NIPS 2014 Workshop on High-energy Physics*

and Machine Learning. Ed. by Glen Cowan et al. Vol. 42. Proceedings of Machine Learning Research. Montreal, Canada: PMLR, 13 Dec 2015, pp. 19–55. URL: `https://proceedings.mlr.press/v42/cowa14.html`.

[2] Pierre Baldi et al. "Parameterized neural networks for high-energy physics". In: *The European Physical Journal C* 76.5 (Apr. 2016). DOI: `10.1140/epjc/s10052-016-4099-4`. URL: `https://doi.org/10.1140%2Fepjc%2Fs10052-016-4099-4`.

[3] Aishik Ghosh, Benjamin Nachman, and Daniel Whiteson. "Uncertainty-aware machine learning for high energy physics". In: *Physical Review D* 104.5 (Sept. 2021). DOI: `10.1103/physrevd.104.056026`. URL: `https://doi.org/10.1103%2Fphysrevd.104.056026`.

[4] Gilles Louppe, Michael Kagan, and Kyle Cranmer. "Learning to Pivot with Adversarial Networks". In: (2016). DOI: `10.48550/ARXIV.1611.01046`. URL: `https://arxiv.org/abs/1611.01046`.

[5] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[6] Byron P. Roe et al. "Boosted decision trees as an alternative to artificial neural networks for particle identification". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 543.2-3 (May 2005), pp. 577–584. DOI: `10.1016/j.nima.2004.12.018`. URL: `https://doi.org/10.1016%2Fj.nima.2004.12.018`.

[7] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: `10.5281/zenodo.3509134`. URL: `https://doi.org/10.5281/zenodo.3509134`.

[8] Zhen Xu et al. "Codabench: Flexible, Easy-to-Use and Reproducible Benchmarking for Everyone". In: *CoRR* abs/2110.05802 (2021). arXiv: `2110.05802`. URL: `https://arxiv.org/abs/2110.05802`.