Training machine learning algorithms on background-contaminated data

M. Borisyak
Higher School of Economics, Moscow, Russia
mborisyak@hse.ru

N. Kazeev[*]
Higher School of Economics, Moscow, Russia
Sapienza University of Rome, Rome, Italy
nikita.kazeev@cern.ch

Abstract

Experimental data in high energy physics is usually contaminated by background. Training a machine learning (ML) algorithm on such data requires accounting for its contribution. A common way of doing so is to use an independent discriminative variable with known distribution (usually the invariant mass) to find the probabilities for each event to be background and then use the sPlot method to subtract its contribution. The method assigns weights to events, some of them negative. For an ML algorithm negative event weights mean that its loss potentially has no lower bound. For some algorithms, like deep neural networks, this leads to diverging training. In this paper, we propose a mathematically rigorous way of training ML algorithms on background-contaminated data. We also conduct experiments that demonstrate the practical feasibility of the method.

1 Introduction

Experimental data obtained in high energy physics experiments usually consists of samples of interest (called signal), which is contaminated by background samples. Moreover, ground-truth labels distinguishing between them are absent; instead, a so-called discriminative variables are often employed as a replacement. In this work, we consider two tasks; the first one is distinguishing between signal and background with absent ground-truth labels. The second one is distinguishing between two signal classes each contaminated by its background, with ground-truth labels indicating signal class are available, however, labels separating signal classes from the corresponding backgrounds are not.

A common technique to filter signal is sideband subtraction. sPlot [15] is a frequently used (reference [15] has over 1460 citations) method for doing background subtraction. It assigns weights (sWeights) to events, which are computed based on known distributions of the control variable, some of the weights are negative. This does not present a problem if the next analysis steps use simple single-dimensional tools, like histograms, but is an obstacle for some multivariate machine learning (ML) methods. In this work, we propose a way to robustly train ML models on data contaminated with background. This being a workshop paper, it presents a combined outlook on our work on the subject, previously published in [3] and [4].

2 Related Work

The practical viability of using ML methods on data weighted with sPlot is demonstrated in [8, 9, 11, 7, 10]. The references, however, do not attempt to analyze the core issue of negative weights impact on ML algorithms, liming themselves to requiring that the classifier supports negative weights.

[*]Corresponding author

Reference [12] proposes classification without labels (CWoLa). This method uses samples with different classes proportions and does not require the per-event probability information. In our experiments, CWoLa shows worse quality than the methods that use this information.

Alternative approaches for filtering signal often involve weak supervision [17, 14, 5]: (almost) unsupervised methods that aim to recover class labels given proportions of classes in the training dataset or groups of samples.

3 sPlot

Take a dataset populated by events from several sources. Assume that the distribution of some variables is known for each source, call these variables discriminative. Usually, the discriminative variable is the reconstructed invariant mass, and the probability densities are estimated by a maximum likelihood fit. sPlot is a statistical technique that reconstructs the distributions of the rest of the variables (call them control), provided they are independent of the discriminative variables for each event source. If the dataset is weighted with weights computed by sPlot (sWeights), the distribution of the control variables will be an unbiased estimate of the distribution of the control variables of the corresponding pure species. Some sWeights are by design negative to subtract the contribution of the background. As an illustration, consider the case, where only background contains the values of the control variable is some interval, e.g., $x \in [0, 1]$. If all weights are positive, the reconstructed distribution will also have positive probability density for $x \in [0, 1]$, which is wrong.

4 The problem of negative event weights

To train a classifier f to separate signal from background using sWeights w_i, one might compute losses on signal samples by applying signal weights w_i, and do the same for background samples with background weights $1 - w_i$:

$$L = \sum_i w_i \log(f(x_i)) + (1 - w_i) \log(1 - f(x_i)). \tag{1}$$

Consider an event with positive signal weight $w > 0$, negative background weight $1 - w < 0$ and the predicted probability of this event being signal p. Then the cross-entropy loss on this sample:

$$\begin{aligned} L &= -w \log(p) - (1 - w) \log(1 - p), & (2)\\ \lim_{p \to 1} L &= -\left(-|1 - w|\right) \lim_{p \to 1} \log(1 - p) = -\infty, & (3) \end{aligned}$$

Directly incorporating sWeights into the cross-entropy loss results in it having no lower bound. The same holds for the mean squared error and other losses without an upper bound in the unweighted case. For some algorithms, such as a large-capacity fully-connected neural network, negative event weights make the optimization problem behind model training ill-defined. An example illustrating diverging training is present in figure 1. The model training using the sWeights as event weights quickly diverges in contrast to the same model training using the true labels or our losses. The model trained on true labels does not even start to overfit – the test score keeps climbing, while the test score of the model trained with sWeights as event weights drops dramatically.

5 Proposed methods

sWeights Averaging (Constrained MSE). Let m be the variable that was used to compute the sWeights, x be the rest of variables. The average of the sWeights over examples with features x equals to the output of the optimal classifier:

$$E_m \left[w(m) \mid x \right] = \frac{p_{\text{signal}}(x)}{p_{\text{signal}}(x) + p_{\text{background}}(x)}. \tag{4}$$

The proof is in appendix A.1. We propose to perform mean-square regression with sWeights as target and control variables as features to estimate the left-hand side. Since the probability lies in $[0, 1]$, one can easily avoid a priori incorrect solutions by, for example, applying the sigmoid function to the model output.

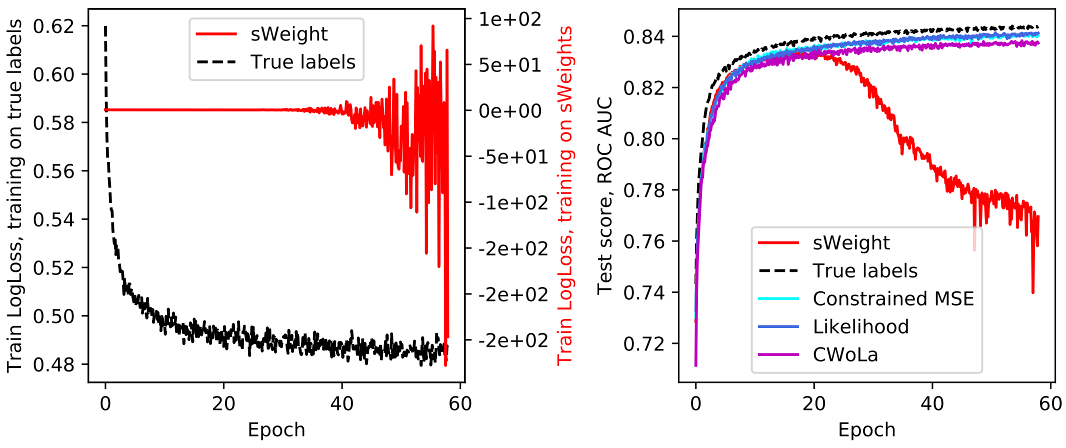


Figure 1: Learning curves of a neural network trained on the UCI Higgs dataset [6, 2] using the true labels and the artificially introduced sWeights. Likelihood and Constrained MSE are described in section 5, CWoLa in [12], the models and experiments details in appendix B.

Exact Maximum Likelihood principle avoids the sPlot technique altogether:

$$L(\theta) = -\sum_i \log\left[f_\theta(x_i) p_{\text{signal}}(m_i) + (1 - f_\theta(x_i)) p_{\text{background}}(m_i)\right], \tag{5}$$

where $f_\theta(x_i)$ is the output of the model; $p_{\text{signal}}(m_i)$ and $p_{\text{background}}(m_i)$ are the probability densities of the signal and background m distributions. Note, that by substituting $p_{\text{signal}}(m_i)$ and $p_{\text{background}}(m_i)$ by the class indicator (0 or 1) in loss (5), it becomes conventional cross-entropy loss. The proof is in appendix A.2.

Classes with Separate Backgrounds. Our approach extends to the case where there are several classes each with its own background defined by a separate set of sWeights; labels are available that indicate to which class signal/background mixture each example belongs. For simplicity, consider the case of two classes. We want to train a model that would be optimal for separating the signal parts of the dataset, ignoring both backgrounds. The loss value of an ML model is the expected value of the per-example loss function. To account the background presence, we can use the sWeights when calculating the expectations:

$$\begin{aligned} L = E_{x,y}\left(E_m\left[\mathrm{w}_1(m) \mid x\right]\right) y \log f(x) + \left(E_m\left[\mathrm{w}_1(m) \mid x\right]\right)(1-y)\log(1-f(x)) \approx \\ \sum_{i=1}^{N} y_i p_1(x_i) \log(f(x_i)) + (1-y_i) p_2(x_i) \log(1-f(x_i)), \end{aligned} \tag{6}$$

where $w_1(m)$ is the sWeight value corresponding to class 1 (signal); $p_1(x)$, $p_2(x)$ are the probabilities that given example is signal. $p_1(x)$ and $p_2(x)$ are estimated by applying our methods for binary classification to the portions of the dataset selected by true label y values.

6 Experimental Evaluation

The UCI Higgs dataset [2, 6] is the largest open dataset from the field of High-Energy Physics. It contains 11 millions samples and 28 features, each representing physics-motivated properties the events. It is labeled and does not feature sWeights, so we introduced them artificially. We assign virtual "mass" distributed independently from the rest of the features to events and use it to compute sWeights. We used neural network and gradient boosting models (as implemented by the CatBoost library [16]). The experiments' details are given in appendix B.1.

The results are present on figures 1 and 2. As expected, training on true labels gives the best performance, and all methods converge on the same quality as the size of the training dataset increases. Constrained MSE and Likelihood loss functions show the same performance, both methods outperform CWoLa. For CatBoost, directly using sWeights as event weights results in stable training and the same performance as for our methods. For neural networks, using sWeights as event weights leads to divergent training. The code of the experiments is published[2].

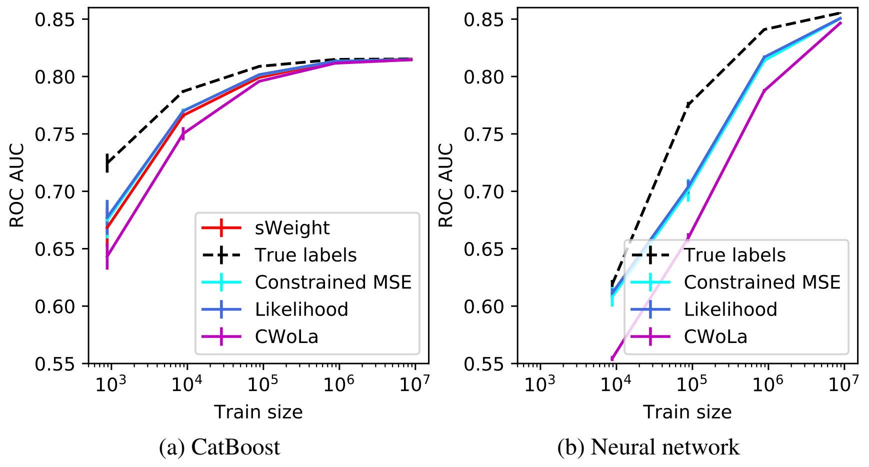


Figure 2: Performance of models using different loss functions on the Higgs dataset as a function of train dataset size. sWeight – cross-entropy weighted with sWeights, it is not reported for the neural network due to divergence of optimization and, hence, highly stochastic nature of the results; True labels – logloss using the true labels; Likelihood and Constrained MSE are described in section 5; CWoLa – method from [12]. Each experiment was repeated 5 times, error bars indicate standard deviation.

The LHCb Muon Identification dataset contains different particle species obtained from different calibration decays [1] and is labeled. Each species has its own background that is subtracted with a separate application of the sPlot method. For simplicity, we use only pion and muon species and ignore the kinematic weights. The dataset has 1.4×10^7 examples and 123 features. The features represent the hits in the muon stations that are the closest to the track along with some high-level variables computed from them.

To our knowledge, this is the only open dataset [13] that has sWeights. Unfortunately, their impact is quite limited, the difference in AUC between a model trained with our methods and a model trained while ignoring the sWeights altogether is around 1% (to our advantage). So we discarded the original sWeights and constructed our own in such a way as to maximize their impact on the classification problem – by artificially introducing into the pionic background a muon-like component.

The experiments' details are given in appendix B.2.

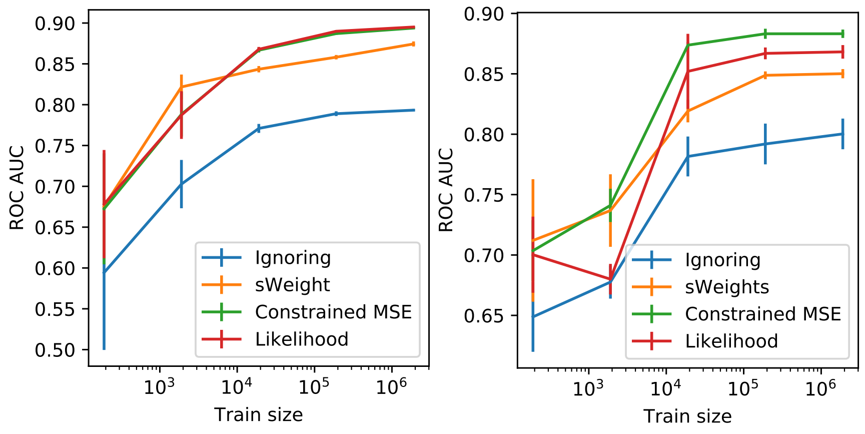
The results are present on figure 3. As expected, the performance of a classifier that ignores the sWeights is significantly lower compared to the rest. In all cases, except for CatBoost with the size of the training dataset $2 \cdot 10^3$, our methods show the best performance.

7 Conclusion

Training machine learning (ML) models on real data (as opposed to Monte-Carlo) is desirable, as it allows the model not to suffer from the imperfection of the simulation. One of the obstacles is that real data often come contaminated with background. A common way of dealing with it, the sPlot method, introduces negative event weights. Training an ML algorithm on a dataset with negative weights means dealing with a loss that potentially has no lower bound. The implications depend on the algorithm in question. In our experiments, neural network training diverges, while gradient boosting decision trees does not.

We propose two loss functions that allow a machine-learning algorithm to obtain class probabilities from background-subtracted data without encountering negative event weight at all. The probability

[2]https://github.com/yandexdataschool/ML-sWeights-experiments/



(a) Catboost (b) Deep Neural Network

Figure 3: Experimental evaluation of our loss functions on the MuID dataset with artificial sWeights. sWeights – using sWeights directly as weights for logloss; Constrained MSE – our regression (5) used to transform the sWeights, the result used as weights for classification; Likelihood – our likelihood (5) for the transformation; Ignoring – training ignoring the sWeights. Each experiment was repeated 5 times, error bars indicate standard deviation.

of an event with given control variables values to be signal is the expected sWeight, which can be estimated by a regression with the corresponding loss function (5). We invoke the Maximum Likelihood principle to construct a loss function that avoids sPlot altogether (5). We implemented our losses for the CatBoost library [16] and published the source code.[3]

Our work paves a rigorous way to use any ML methods on data contaminated by background that can be described by independent discriminative variables.

References

[1] Roel Aaij et al. Selection and processing of calibration samples to measure the particle identification performance of the LHCb experiment in Run 2. *EPJ Techniques and Instrumentation*, 6(1):1, 2019.

[2] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

[3] M. Borisyak and N. Kazeev. Machine learning on data with sPlot background subtraction. *Journal of Instrumentation*, 14(08):P08020–P08020, aug 2019.

[4] M. Borisyak and N. Kazeev. Machine learning on sWeighted data. In *submitted to Journal of Physics: Conference Series (ACAT-2019 proceedings)*. IOP Publishing, 2019.

[5] Lucio Mwinmaarong Dery, Benjamin Nachman, Francesco Rubbo, and Ariel Schwartzman. Weakly supervised classification in high energy physics. *Journal of High Energy Physics*, 2017(5):145, 2017.

[6] Dheeru Dua and Casey Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[7] Andreas Hoecker et al. TMVA – Toolkit for multivariate data analysis. *arXiv preprint physics/0703039*, 2007.

[8] Thomas Keck. *Machine learning algorithms for the Belle II experiment and their validation on Belle data*. PhD thesis, KIT, Karlsruhe, 2017.

[3]https://github.com/kazeevn/catboost/tree/constrained_regression

[9] Benjamin Lipp. sPlot-based training of multivariate classifiers in the Belle II analysis software framework. *Bachelor thesis, KIT*, 2015.

[10] Claudia Marino. X_b Search and Measurement of the $Y(1)$, $Y(2S)$ and $Y(3S)$ Polarization. Technical report, Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), 2009.

[11] D Martschei, M Feindt, S Honc, and J Wagner-Kuhr. Advanced event reweighting using multivariate analysis. In *Journal of Physics: Conference Series*, volume 368, page 012028. IOP Publishing, 2012.

[12] Eric M Metodiev, Benjamin Nachman, and Jesse Thaler. Classification without labels: Learning from mixed samples in high energy physics. *Journal of High Energy Physics*, 2017(10):174, 2017.

[13] Higher School of Economics. International data analysis olympiad. `https://idao.world/`, 2019.

[14] Giorgio Patrini, Richard Nock, Paul Rivera, and Tiberio Caetano. (almost) no label no cry. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 190–198. Curran Associates, Inc., 2014.

[15] Muriel Pivk and Francois R Le Diberder. sPlot: a statistical tool to unfold data distributions. *NIMA*, 555(1-2):356–369, 2005.

[16] Liudmila Prokhorenkova et al. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, pages 6638–6648, 2018.

[17] Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(Oct):2349–2374, 2009.

A Proofs

A.1 Constrained MSE

Let m be the variable that was used to compute the sWeights, x is the rest of variables. It is possible to weight the dataset, so x distribution will be that of the signal component with positive weights equal to class probabilities $w(x) = \frac{p_{\text{signal}}(x)}{p_{\text{mix}}(x)}$. The data weighted with the sWeights achieve the same distribution. Let m be the variable that was used to compute the sWeigths, x be the rest of variables and $f(x)$ is any smooth function of x.

$$E_{x\sim p_{\text{sig}}}(f(x)) = \int dx f(x) p_{\text{sig}}(x). \tag{7}$$

Define

$$W(x) = \frac{p_{\text{sig}}(x)}{p_{\text{mix}}(x)}. \tag{8}$$

Then

$$E_{x\sim p_{\text{sig}}}(f(x)) = \int dx f(x) W(x) p_{\text{mix}}(x) \tag{9}$$

By definition of sWeights:

$$E_{x\sim p_{\text{sig}}}(f(x)) = \int dx dm \cdot w(m) f(x) p_{\text{mix}}(x, m), \tag{10}$$

where $w(m)$ is the sWeight.

$$E_{x\sim p_{\text{sig}}}(f(x)) = \int dx dm \cdot w(m) f(x) p_{\text{mix}}(x) p_{\text{mix}}(m|x) \tag{11}$$

Rearrange the multipliers in the double integral:

$$E_{x\sim p_{\text{sig}}}(f(x)) = \int dx f(x) p_{\text{mix}}(x) \int dm w(m) p_{\text{mix}}(m|x) \tag{12}$$

From equation 9,

$$\int dx f(x) W(x) p_{\text{mix}}(x) = \int dx f(x) p_{\text{mix}}(x) \int dm w(m) p_{\text{mix}}(m|x). \tag{13}$$

Therefore,

$$W(x) = \int dm w(m) p_{\text{mix}}(m|x) = \mathbb{E}_m(\text{sWeight}(x, m)) \tag{14}$$

A.2 Exact Maximum Likelihood

Denote signal and background classes as S and B, model parameters as θ. By definition of the log-likelihood $l(\theta)$:

$$l(\theta) = \sum_i \log p(x_i, m_i \mid \theta) = \sum_i \log \left[p(x_i, m_i \mid \theta, \text{S}) P(\text{S}) + p(x_i, m_i \mid \theta, \text{B}) P(\text{B}) \right]. \tag{15}$$

Since x_i and m_i are assumed to be independent within individual classes, expression (15) can be simplified further:

$$\begin{aligned} l(\theta) = & \\ & \sum_i \log \left[p(x_i \mid \theta, \text{S}) p(m_i \mid \text{S}) P(\text{S}) + p(x_i \mid \theta, \text{B}) p(m_i \mid \text{B}) P(\text{B}) \right] = \\ & \sum_i \log \left[P(\text{S} \mid x_i, \theta) p(m_i \mid \text{S}) p(x_i) + p(\text{B} \mid x_i, \theta) p(m_i \mid \text{B}) p(x_i) \right] = \\ & \qquad \sum_i \log \left[P(\text{S} \mid x_i, \theta) p(m_i \mid \text{S}) + P(\text{B} \mid x_i, \theta) p(m_i \mid \text{B}) \right] + \text{const}, \end{aligned} \tag{16}$$

which leads to the following loss function:

$$L(\theta) = -\sum_i \log \left[f_\theta(x_i) p_{\text{signal}}(m_i) + (1 - f_\theta(x_i)) p_{\text{background}}(m_i) \right], \tag{17}$$

where $f_\theta(x_i)$ is the output of the model; $p_{\text{signal}}(m_i)$ and $p_{\text{background}}(m_i)$ are the probability densities of the signal and background m distributions.

B Experiments parameters

B.1 Higgs

The neural networks have 3 hidden layers: 128, 64, 32 neurons for the experiments with $8.8 \cdot 10^6$ and $8.8 \cdot 10^5$ samples in the training sets, and 64, 32, 16 neurons in cases of $8.8 \cdot 10^4$ and $8.8 \cdot 10^3$ training samples. Model capacity varies to adjust for the low sample sizes.

All networks use leaky ReLu (with leakiness 0.05) activation function. Networks are optimized by adam algorithm with learning rate $2 \cdot 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ for $2.2 \cdot 10^6$ steps (32 full 'passes' over the original training sample) with batch size 128; learning rate is set to the value lower than commonly used ones due to high variance of the gradients for CWoLa.

Each experiment is repeated 5 times with varying training datasets (if subsampled) and initial weights, however, within each experiment networks for different methods have identical conditions: they share initial weights, are trained on the same subsample and use identical sequences of batches.

Due to the presence of logarithm, loss function (5) might become computationally unstable for networks with large weights. For the experiments with $8.8 \cdot 10^4$ and fewer training samples, l_2 regularization is introduced to the exact maximum likelihood loss function. This regularization does not affect results in any significant way, besides limiting network weights to computationally stable values. A similar effect can be achieved by introducing a small constant to all values under logarithms.

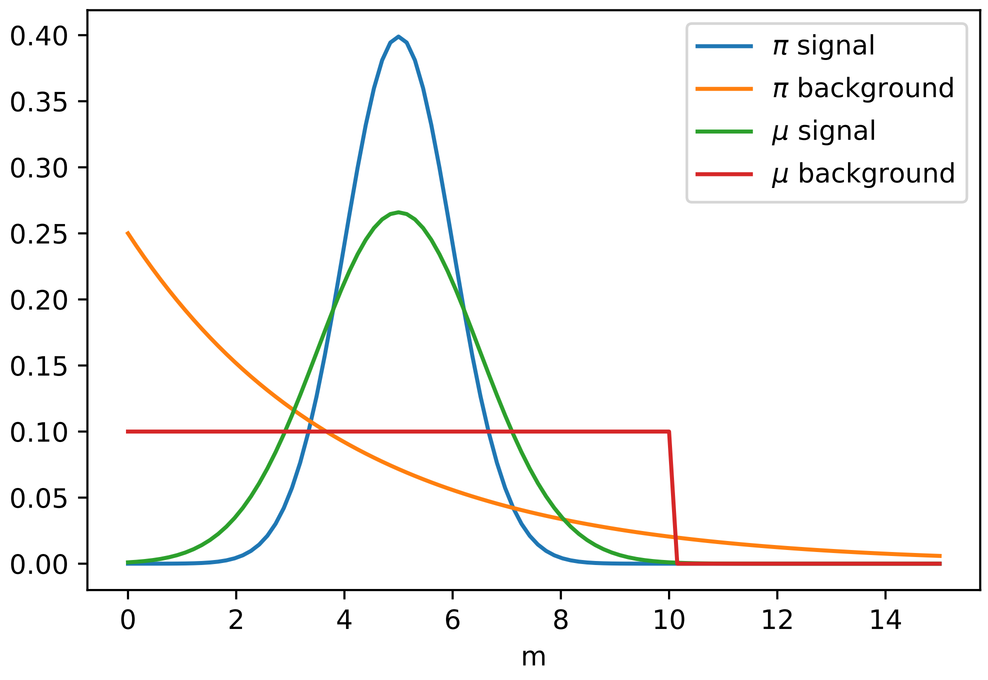


Figure 4: Discriminative variable distributions used to regenerate the sWeights for the additional test.

For gradient boosting experiments we use CatBoost with the following parameters: 1000 trees, leaf_estimation_method="Gradient", version 0.10.2 with our losses added and check for negative weights removed, source code is available [4]. Each experiment is repeated 10 times with varying training dataset (if subsampled).

B.2 Muon ID

The sWeights generation procedure First, we trained a CatBoost classifier to separate signal and background. We calibrated its output and used it as event weights to train a classifier to separate muons and pions as proposed in section 5. We selected 30% of examples with muon label with the highest scores and marked them as pionic background. Next, we assigned definite signal and background labels by cutting on the signal probabilities obtained from the first step. We generated "pseudomass" for both muon and pion background, which we used to compute the new sWeights.

The models' parameters:

- Fully-connected neural networks (NN): 4 layers, 512, 256, 128 neurons in the first three layers and either 1 or 2 neurons in the last one, leaky ReLu (0.05), optimized by Adam algorithm with learning rate $2 \cdot 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, trained for 32 epochs; regressors and classifiers use the same architecture; Each experiment is repeated 5 times with varying training dataset (if subsampled) and initial weights, however, within each experiment networks for different methods have identical conditions: they share initial weights, are trained in the same subsample and use an identical sequence of batches.
- CatBoost: 500 trees, leaf_estimation_method="Gradient", version 0.10.2 with our losses added and check for negative weights removed. Each experiment is repeated 10 times with varying training dataset (if subsampled).

[4]anonymized url