Machine Learning on data with sPlot background subtraction

M. Borisyak[a] **N. Kazeev**[a,b,1]

[a]*Laboratory of Methods for Big Data Analysis, National Research University Higher School of Economics, 11, Pokrovsky boulevard, Moscow 109028, Russia*

[b]*Sapienza University of Rome, Piazzale Aldo Moro, 5, 00185 Roma RM, Italy*

E-mail: nikita.kazeev@cern.ch

ABSTRACT: Data analysis in high energy physics often deals with data samples consisting of a mixture of signal and background events. The sPlot technique is a common method to subtract the contribution of the background by assigning weights to events. Part of the weights are by design negative. Negative weights lead to the divergence of some machine learning algorithms training due to absence of the lower bound in the loss function. In this paper we propose a mathematically rigorous way to train machine learning algorithms on data samples with background described by sPlot to obtain signal probabilities conditioned on observables, without encountering negative event weight at all. This allows usage of any out-of-the-box machine learning methods on such data.

---

[1]Corresponding author.

Contents

1   Introduction

Experimental data obtained in high energy physics experiments usually consists of contributions from different event sources. We consider the case, where the distribution of some variables is known for each source and call these variables discriminative. Usually, the variable is the reconstructed invariant mass, and the probability densities are estimated by a maximum likelihood fit. The distribution of other (control) variables also presents interest, as an analysis quality check or a training sample of a machine learning algorithm. The sPlot technique [1] allows to reconstruct the distribution of the control variables, provided they are independent of the discriminative variables. sPlot assigns weights (sWeights) to events, some of them negative. This does not present a problem for simple one-dimensional analysis tools, like histograms, but is an obstacle for multivariate machine learning methods that require the loss function to be bounded from below.

This paper is structured as following. In section 2 we briefly introduce the sPlot technique. In section 3 we describe the literature concerning machine learning on data weighted with the sPlot.

In section 4 we discuss the implications of negative event weights on training of machine learning algorithms. In section 5 we propose methods to robustly obtain class probabilities conditioned on the control variables. In section 6 we present experimental results that demonstrate practical viability of the proposed method.
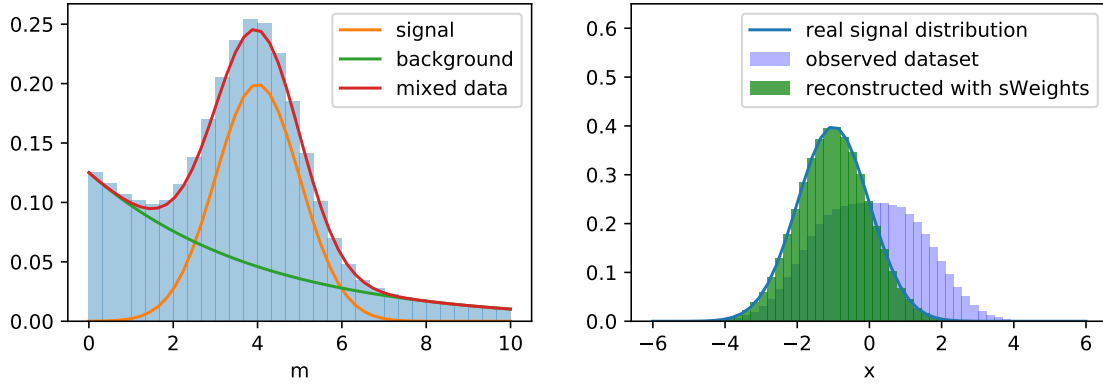
2   sPlot

The full derivation of sPlot is present in [1]. Here we briefly recite the key formulas. Take a dataset populated by events from N_s sources. Let $p_k(m)$ be probability density function of the discriminative variable m of the i-th species; N_k the number of events expected on the average for the k-th species; N – the total number of events; m_e – the value of m for the e-th event. Define matrix $\mathbf{V}$:

$$\mathbf{V}_{nj}^{-1} = \sum_{e=1}^{N} \frac{p_n(m_e) p_j(m_e)}{\left(\sum_{k=1}^{N_s} N_k p_k(m_e) \right)^2} \tag{2.1}$$

The sWeight for the e-th event corresponding to the n-the species is obtained using the following transformation:

$$\text{sWeight}_n(m_e) = \frac{\sum_{j=1}^{N_s} \mathbf{V}_{nj} p_j(m_e)}{\sum_{k=1}^{N_s} N_k p_k(m_e)} \tag{2.2}$$

If the dataset is weighted with sWeights, the distribution of the control variables will be an unbiased estimate of the corresponding pure species. In the rest of the paper we deal with the two species scenario, named signal and background. An example of sPlot application is present on figure 1.

Figure 1: An example of sPlot application. To the left: the known distributions of m. To the right: the mixture and reconstructed distribution of x.

3   Related work

There are several works concerning machine learning and sWeights: [2–5]. They propose a training procedure for the case where a classifier is desired to separate the same signal and background that are defined by the sPlot. Take each event twice, once as signal, once as background with the

corresponding sWeights, then train the classifier as usual. The works also demonstrate practical viability of using machine learning methods on data weighted with sPlot. They, however, do not attempt to analyze the core issue of negative weights impact on machine learning algorithms, liming themselves to requiring that the classifier supports negative weights.

Another possible approach is suggested by [6] – classification without labels (CWoLa). It is possible to use the discriminative variable to define regions with different signal/background proportions, then train a classifier to separate them, without using any weights or true labels. The authors prove that the optimal classifier for this case would also be the optimal classifier for signal against background. This method ignores the per-event probability information. In our experiments (presented in section 6), CWoLa has worse quality on a real finite dataset than the methods that use this information.

4 The problem of negative weights

Let us consider an event with positive signal weight $w_s > 0$ and negative background weight $w_b < 0$ and the classic cross-entropy loss function:
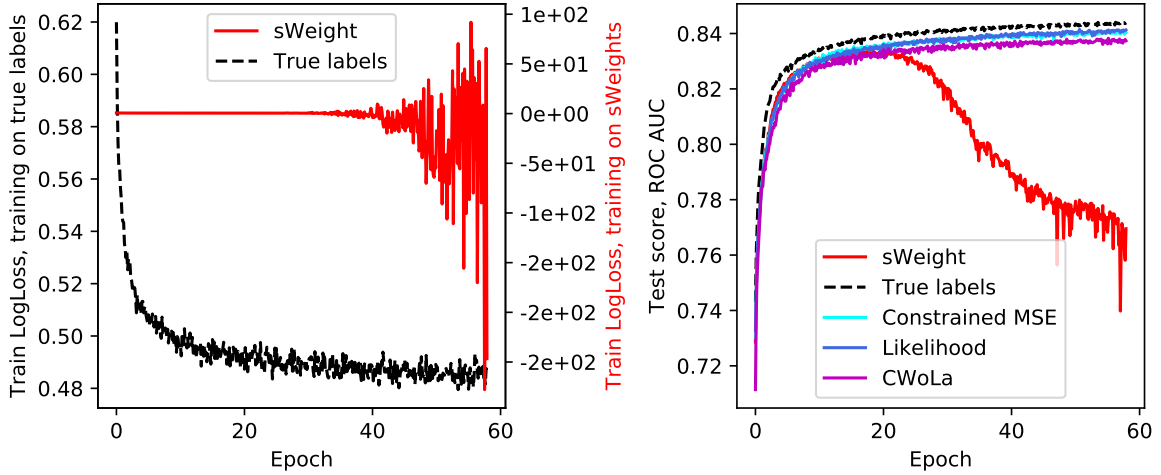
$$L = -w_s \log(p_s) - w_b \log(1 - p_s), \tag{4.1}$$

where p_s is the model output – the predicted probability of this event being signal.

$$\lim_{p_s \to 1} L = -(-|w_b|) \lim_{p_s \to 1} \log(1 - p_s) = -\infty. \tag{4.2}$$

Thus, directly incorporating sWeights into the cross-entropy loss causes it to lose the lower bound. The same holds for the mean squared error and other losses without an upper bound in the unweighted case. Training most machine learning algorithms is an optimization problem, and, for some algorithms, such as a large-capacity fully-connected neural network, negative event weights make this optimization problem ill-defined, as the underlying optimization target loses the lower bound as well. An example illustrating diverging training is present on figure 2. 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. As expected, the test ROC AUC for the model that was trained using the true labels is the best among all methods.

However, most machine learning algorithms do not blindly minimize the loss value on the training dataset, as this is likely to lead to overfitting. They add various regularization terms to the optimized functional that, in general, penalize model complexity or overconfidence. We are not aware of peer-reviewed literature exploring the impact of regularization on learning with negative weights. One such technique appears in discussions within the High Energy Physics community [7]. They propose avoiding the unbounded loss by requiring leafs of decision tree to have positive total weight. It is also possible to regularize neural networks into having bounded loss. For example, by using the L2 regularization on weights and taking the root of the degree equal to the number of layers plus one from the cross-entropy loss.

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

Nevertheless, it is unclear how negative weights combined with such regularization would affect classifier performance. A more detailed study falls outside of the scope of this paper, which presents a principled approach to avoid the problem entirely.

5   Our approaches

5.1   sWeights averaging (Constrained MSE)

Let m be the variable that was used to compute the sWeigths, x the rest of the variables. It can be shown that:

$$\mathbb{E}_m\left[w(m) \mid x\right] = \frac{p_{\text{signal}}(x)}{p_{\text{mix}}(x)}. \tag{5.1}$$

The formal proof is available in appendix A.1. Notice, that the right-hand side of (5.1) is the optimal output of a classifier, while the left-hand side can be estimated by a regression model. Our first proposed approach is to perform mean-square regression directly on sWeights. Since the optimal output lies in $[0, 1]$, one can easily avoid a priori incorrect solutions by, for example, applying sigmoid function to the model output. The resulting loss function is the following:

$$L = \sum_i \left(w_i - \frac{e^{f_\theta(x_i)}}{1 + e^{f_\theta(x_i)}}\right)^2, \tag{5.2}$$

where w_i is the sWeight and $f_\theta(x_i)$ is the model output. This loss has been implemented for the CatBoost machine learning library [8] and is available on GitHub.[1]

---

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

5.2 Exact maximum likelihood

Alternatively, one can invoke Maximum Likelihood principle and avoid the sPlot technique altogether. This leads to the following loss function (derivation is in A.2):

$$L(\theta) = -\sum_i \log \left[f_\theta(x_i) p_{\text{signal}}(m_i) + (1 - f_\theta(x_i)) p_{\text{background}}(m_i) \right], \tag{5.3}$$

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 $p_{\text{signal|background}}(m_i)$ by the class indicator ($y_i = 1$ if x_i is a signal sample, $y_i = 0$ otherwise) in loss (5.3), a conventional expression for cross-entropy loss can be obtained.

6 Experimental evaluation

To demonstrate viability of our methods on practical problems, we tested them on the UCI Higgs dataset [9, 10]. We used neural network and gradient boosting models, their detailed description is in B. The dataset is the largest open dataset from the field of High-Energy Physics. It has 28 tabular features. We split it into train and test parts containing $8.8 \cdot 10^6$ and $2.2 \cdot 10^6$ events respectively. The dataset is labeled and it does not feature sWeights, so we introduced them artificially. For both signal and background events we added a virtual "mass" distributed as shown on figure 1 and used it to compute sWeights. We also compared with the CWoLa method [6] by splitting the data into a signal-majority and a background-majority regions. The signal region was chosen with center at $m = 4$ (center of the signal m distribution) and width so that it would include half of the events.

The results are present on figures 2 and 3. As expected, training on true labels gives the best performance. 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 in our methods. For neural networks, using sWeights as event weights leads to divergent training. The performance difference between the methods gets smaller with train dataset size increase. The code of the experiments is available on GitHub.[2]
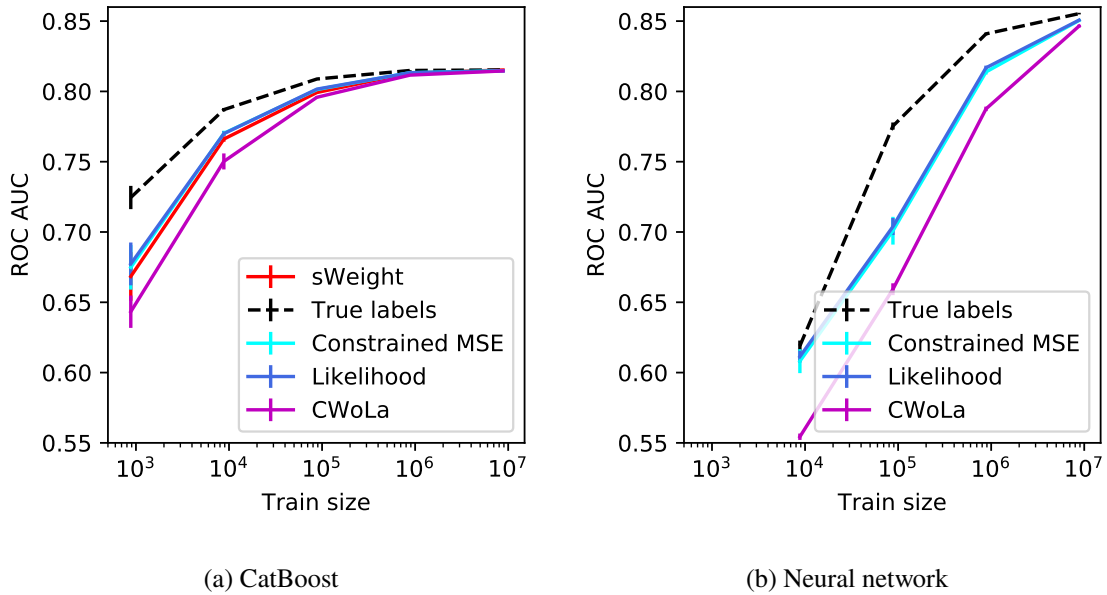
7 Conclusion

Training a machine learning 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 over oblivious decision trees does not.

Our contribution is the 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 of an event with given control variables values to be signal is the expected sWeight, that can be estimated by a regression with the corresponding loss function (5.1). We invoke the Maximum Likelihood principle to construct a loss function that avoids sPlot and associated problems with negative event weights (5.2).

---

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

(a) CatBoost   (b) Neural network

Figure 3: Experimental evaluation of performance of 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; Constrained MSE – our loss function defined by (5.2); Likelihood – our loss function defined by (5.3); CWoLa – method from [6];

Our work paves a rigorous way to use any machine learning methods on data with sPlot-based background subtraction.

Acknowledgments

References

[1] M. Pivk and F. R. Le Diberder, *splot: A statistical tool to unfold data distributions*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **555** (2005) 356–369.

[2] T. Keck, *Machine learning algorithms for the Belle II experiment and their validation on Belle data*. PhD thesis, KIT, Karlsruhe, 2017. 10.1007/978-3-319-98249-6.

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

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

[5] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss et al., *TMVA – Toolkit for multivariate data analysis*, *arXiv preprint physics/0703039* (2007) .

[6] E. M. Metodiev, B. Nachman and J. Thaler, *Classification without labels: Learning from mixed samples in high energy physics*, *Journal of High Energy Physics* **2017** (2017) 174.

[7] "GitHub issue: Ensemble models (and maybe others?) don't check for negative sample_weight, last accesed: 2019-06-01." https://github.com/scikit-learn/scikit-learn/issues/3774.

[8] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush and A. Gulin, *CatBoost: unbiased boosting with categorical features*, in *Advances in Neural Information Processing Systems*, pp. 6638–6648, 2018.

[9] P. Baldi, P. Sadowski and D. Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, *Nature communications* **5** (2014) 4308.

[10] D. Dua and C. Graff, *UCI machine learning repository*, 2017.

[11] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014) .

A Proofs

A.1 Constrained MSE

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.

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

define

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

Then

$$E_{x \sim p_{\text{sig}}}(f(x)) = \int dx f(x) W(x) p_{\text{mix}}(x) \tag{A.1}$$

By definition of sWeights:

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

where $w(m)$ is the sWeight.

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

Rearrange the multipliers in the double integral:

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

From equation A.1

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

Therefore,

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

A.2 Exact maximum likelihood

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

$$l(\theta) = \sum_i \log p(x_i, m_i \mid \theta) = \sum_i \log \left[p(x_i, m_i \mid \theta, \mathrm{S}) P(\mathrm{S}) + p(x_i, m_i \mid \theta, \mathrm{B}) P(\mathrm{B}) \right]. \quad \text{(A.2)}$$

Since x_i and m_i are assumed to be independent within individual classes, expression (A.2) can be simplified further:

$$l(\theta) =$$

$$\sum_i \log \left[p(x_i \mid \theta, \mathrm{S}) p(m_i \mid \mathrm{S}) P(\mathrm{S}) + p(x_i \mid \theta, \mathrm{B}) p(m_i \mid \mathrm{B}) P(\mathrm{B}) \right] =$$

$$\sum_i \log \left[P(\mathrm{S} \mid x_i, \theta) p(m_i \mid \mathrm{S}) p(x_i) + p(\mathrm{B} \mid x_i, \theta) p(m_i \mid \mathrm{B}) p(x_i) \right] =$$

$$\sum_i \log \left[P(\mathrm{S} \mid x_i, \theta) p(m_i \mid \mathrm{S}) + P(\mathrm{B} \mid x_i, \theta) p(m_i \mid \mathrm{B}) \right] + \mathrm{const}, \quad \text{(A.3)}$$

which leads to the following loss function:

$$L(\theta) = -\sum_i \log \left[f_\theta(x_i) p(m_i \mid \mathrm{S}) + (1 - f_\theta(x_i)) p(m_i \mid \mathrm{B}) \right] \quad \text{(A.4)}$$

B Models parameters

B.1 Neural Networks

For the experiments with fully-connected neural networks we use networks with 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 (0.05) activation function. Networks are optimized by adam [11] 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 dataset (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 sequence of batches.

Due to use of log function, loss function (5.3) might become computationally unstable for networks with large weights. For the experiments with $8.8 \cdot 10^4$ 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.

B.2 CatBoost

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: `https://github.com/kazeevn/catboost/tree/constrained_regression`. Each experiment is repeated 10 times with varying training dataset (if subsampled).