Exploring Machine Learning for Particle Physics

Aristide Baratin[*] Shawn Tan[*]

Pierre-Andre Brousseau Anirudh Goyal Alex Lamb

Abstract

In this article, we report[2] our work on the Kaggle Challenge: Flavours of Physics: Finding $\tau^- \to \mu^+\mu^-\mu^-$. The main goal of this challenge is to develop powerful classifiers for the detection of 'new physics' – specifically, violation the lepton flavour conservation guaranteed by the standard model – in the Large Hadron Collider (LHC). We build several models for the challenge, which include boosted decision trees and neural networks, and report our findings on their performance. We include a cautionary tale of how the use of some specific features of the data can dramatically impact the physical relevance of the results.

1 Introduction

1.1 New physics in the LHC

The standard model for particle physics, which results from decades of collaboration between the 30s and the end of the 70s, provides a complete and remarkably accurate description of matter particles and their interactions at microscopic scales Ramond (1999). The model is still undefeated after years of intense experimentation in the successive generations of particle colliders. The latest success to date is the experimental discovery in 2012 in the LHC of the Higgs boson, introduced in the standard model in 1964.

However despite any major conflict with experiments so far, physicists have good reasons to believe that the standard model is not the end of the story. It does not explain neutrino oscillations, dark matter, nor takes into account the gravitational interaction. One of the main challenges in experimental particle physics has been to detect signals of new physics - i.e phenomena that would conflict with the rules of standard model. The $\tau^- \to \mu^+\mu^-\mu^-$ phenomenon is one such case, and this is the subject of the challenge. It violates a rule of the standard model: the conservation of a physical quantity called the lepton flavour – the number of electrons and electron-neutrinos, muons and muon-neutrinos, and tau and tau-neutrinos. Interestingly, in many proposed extensions of the standard model, this conservation law is absent, so that decays such as $\tau^- \to \mu^+\mu^-\mu^-$ are possible. Hence the observation of such event would be a clear sign of new physics beyond the standard model.

1.2 Why machine learning?

The outcome of an experiment searching for signals of a new phenomenon results from a binary classification task: to disentangle the signal from ordinary physics 'background'. More precisely, the eventual goal is to discriminate between two hypothesis:

background only V.S signal + background

A major difficulty in the case of events such as the $\tau^- \to \mu^+\mu^-\mu^-$ decay, where a single particle splits into three, is that they are *rare* – that is, even in a context where lepton flavour is allowed, the

[*]Equal contribution.

[2]Our code source can be found in https://github.com/ArisBar/IFT-6390-project

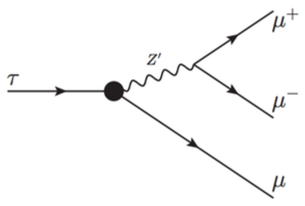

Figure 1: Decay violating lepton flavour symmetry of the standard model

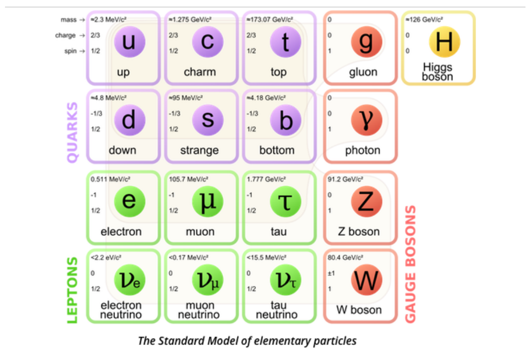

Figure 2: Particles of the Standard Model.

decay occurs with extremely low probability. This means the ratio background/signal is extremely high. In this context the optimization of machine learning algorithms to obtain powerful discriminators is crucial, and has a direct impact on the quality of the results.

1.3 Outline

The rest of the report is structured as follows: In Section 2, we further describe the nature of the data as well the evaluation metric provided for the challenge. In Section 3, we present our first results, with the baseline algorithms we have developed for this data. In Section 4, we point out a subtlety with the simulated data, which raises the question of the physical interpretability of the results. We discuss additional tests that came along with the challenge, whose purpose is precisely to circumvent this problem. Finally in Section 5, taking into account these new constraints, we propose a new approach using deep learning techniques, which allows us to obtain a very good performance while arguably remaining physically sound.

2 The data

2.1 Real vs simulated data

The dataset is a labeled dataset (the label being 1 for signal and 0 for background events), which contains *both* Monte-Carlo simulated signal data and real background data from the LHCb. The reason why the whole dataset is not simulated is the lack of the computing power needed to produce the enormous amount of background samples required to model the event. A major difficulty arises from such a mixing of real and simulated data: a classifier could learn the imperfections of the simulation instead of actually discriminating signal and background. We will come back to this point in Section 4.

2.2 Features

The dataset consists of 67553 rows, with 45 usable features. It is essentially a binary classification problem, with additional constraints, and 61.7% of the datapoints given correspond to simulated signal data.

Table 1: Details about the dataset.

Number of rows:	67553
Columns (including id):	51
Columns useful at test time:	45
Ratio of $+$ve examples to total:	0.617

A first step for us has been to clean the data (removing the features that should not be used for training or imposing constraints on others, as recommended by the challenge team) and to gain an understanding of the features of the dataset. These include two main categories of features:

- **Kinematical** features such as the energy and momenta of the outgoing muon particles, expressed in a suitable coordinate system.
- **Geometrical** features such as the impact parameter and various angles characterizing the trajectory of the τ particle, as well isolation variables, explained in Chrzaszcz *et al.* (2015).

Its important to emphasize that the only ‘observed’ particles in the process are the three outgoing muon particles. The tau particle shows up as an intermediate process, as a result of a proton collision; it is thus never directly observed and its properties can only be inferred from the actual observations. In fact, using elementary notions of special relativity such as for example the conservation of total energy-momenta for the event,

$$\mathbf{p}_\tau = \sum_{i=1}^{3} \mathbf{p}_{\mu_i}$$

additional features for the tau candidate can be calculated and possibly added to the dataset. We thus created 7 new features including the reconstructed momenta, speed, energy and mass of the τ particle. We added those features in the file features.py of our source code.

2.3 Evaluation

The evaluation metric for this challenge is a weighted Area-Under-Curve (AUC), for the receiver operating characteristic (ROC). The ROC curve is obtained by adjusting the threshold on the output of the model. A high threshold results in a low False Positive Rate (FPR), and a low threshold gives a True Positive Rate (TPR). Plotting several values at different thresholds, we get a graph much like the one below[3].

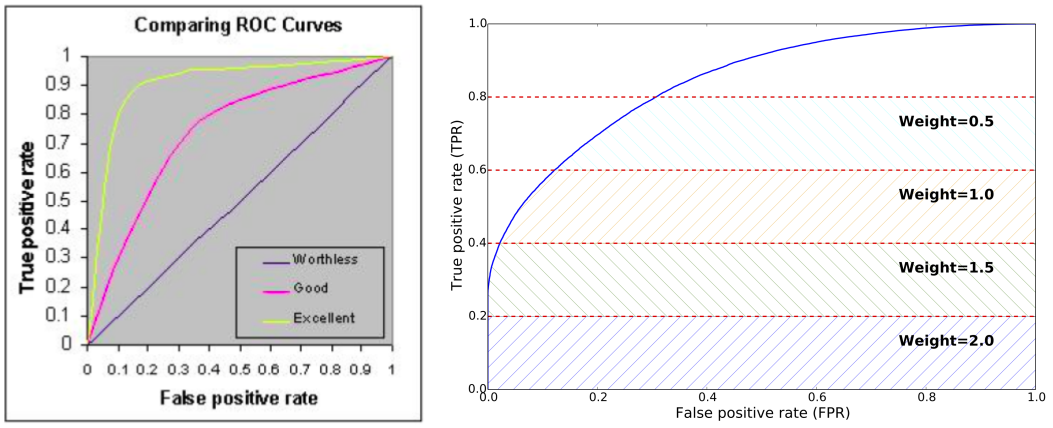

Figure 3: Left: Comparing ROC curves, Right: weighted ROC evaluation metric.

The slight difference in this metric from the standard AUC is that it is weighted differently at different bands, which were chosen to match the evaluation methodology used by CERN scientists.

[3]Courtesy of http://gim.unmc.edu/dxtests/roc3.htm

3 First results

We present our first set of algorithms: as baseline algorithms, we have implemented a naive Bayes classifer and a decision tree. We then moved to boosted trees implemented with Adaboost. All models have been implemented using the library scikit-learn. In Section 5 we will present results obtained with neural networks.

Throughout this section we trained and tested our models using k-fold cross-validation using $k = 5$ equal sized subsamples[4]. We present the results obtained on the original dataset, with no additional features.

3.1 Bayes classifiers

As first baseline algorithm, we implemented a Gaussian naive Bayes classifier. The naive Bayes classifier is a very simple algorithm that assumes features as independent given the class and that no hidden or latent variables influence the prediction process John and Langley (1995). Despite its simplicity, it can be in practice very successful and competitive with more sophisticated classifiers Rish (2001). In this case, using the Gaussian naive Bayes method, the values for each of the features are assumed normally distributed within each class. The parameters for these Gaussian distributions are as such learned by the model.

For the validation, we obtain an **AUC of 0.9613**.

3.2 Decision Tree Classifier

Our second baseline algorithm is a decision tree. Decision tree classifiers are commonly used as an alternative to artificial neural networks in particle identification. They aim to model the classification problem as a system of leaves and nodes where each leaf represents a targeted class given the path from the root to the said leaf.

In a more detailed way, given a splitting value for a variable, the set of events are split in two parts according to the splitting value. The selected variable and splitting value is chosen as to give the best separation between the two classes and is defined as a node. The initial set of events is now split into two branches. Each branch will now be treated as a new set of events where the process of selecting a splitting value and a variable will occur leading to more nodes and more branches. One can keep these steps of splitting until the set of events is of a single class, pure. Other stopping criteria can be added. When the set of events reaches these one of these criteria it will be automatically established as a leaf. Some of these stopping criteria are the maximum allowed depth of the built tree, the minimal amount of events in a set and the target purity for a set Roe *et al.* (2005).

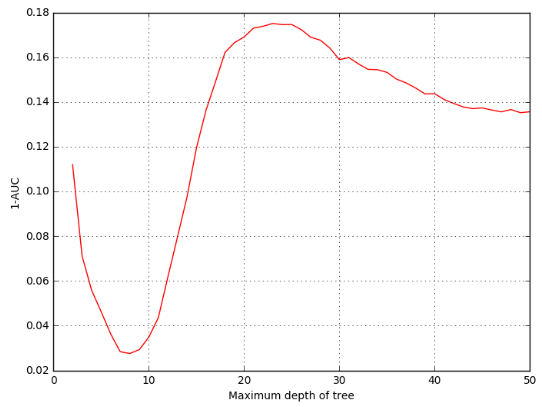

Figure 4: Plot of the cross validation error against the max-depth of the decision tree.

[4]Note that, although we did not do this, in principle we should have also extracted a subset of the data beforehand allowing us to compare the performance of all our models, including the neural networks of Section 5, on the same test set, in an unbiased fashion.

Following the validation as shown in figure 3, we obtain an **AUC of 0.9725** for the best hyper-parameter value of: 6 for the maximum depth.

3.3 Boosted Trees

Boosting methods enable the reduction of bias by taking a set of weak classifiers and building a strong classifier. In this case, the set of weak classifier is a set of very shallow decision tree classifiers called stumps and are obtained by limiting the maximum depth stopping criteria to a very low value.

During the selection of the splitting value and variable, every event are considered equal and it is desired to maximize the purity of each branch according to the split. However, in weighted events, not all of the events are considered of equal value so the selection of splitting value and variable is influenced by the weights.The first step of the boosting method is creating a weak tree classifier and identify miss-classified samples out of the training set where all the events are weighted equally. The second step, is to again create a weak tree classifier while this time giving more weight to the miss-classified samples of the previous classifier. The resulting classifier after to steps is a weighted average of the two classifiers. Trees are then added sequentially by increasing the weights of the miss-classified events after each step Roe *et al.* (2005).

In the AdaBoost algorithm, the weight of a tree T_m is given by α_m and computed as follow.

$$\begin{aligned} err_m &= \frac{\sum_{i=1}^{N} w_i I(y_i \neq T_m(x_i))}{\sum_{i=1}^{N} w_i} \\ \alpha_m &= ln(\frac{1 - err_m}{err_m}) \end{aligned} \tag{1}$$

The weights of each event w_i is then adjusted.

$$w_i \leftarrow wi * e^{\alpha_m I(y_i \neq T_m(x_i))} \tag{2}$$

The resulting class for each event x_i is given by $T(x_i)$.

$$T(x_i) = \sum_{m=1}^{N_{tree}} \alpha_m T_m(x_i) \tag{3}$$

Following the validation error against the number of trees, for various (small) values of the maximum depth as shown in figure 3.3, we obtain an **AUC of 0.9872** for the best hyper-parameter value of: 2 for the maximum depth and 83 for the number of trees.

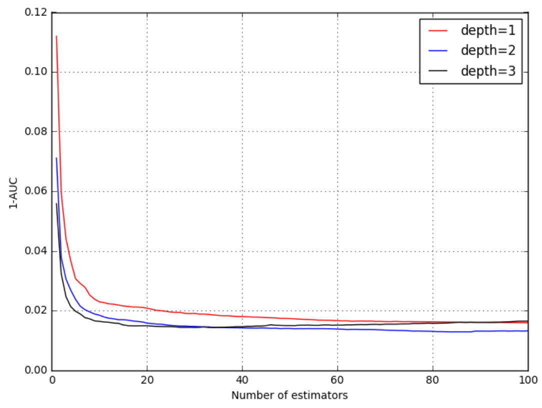

Figure 5: Cross validation error against the number of trees for different values of the depth.

We now move to a discussion of additional requirements that our classifiers must satisfy to be physically sound.

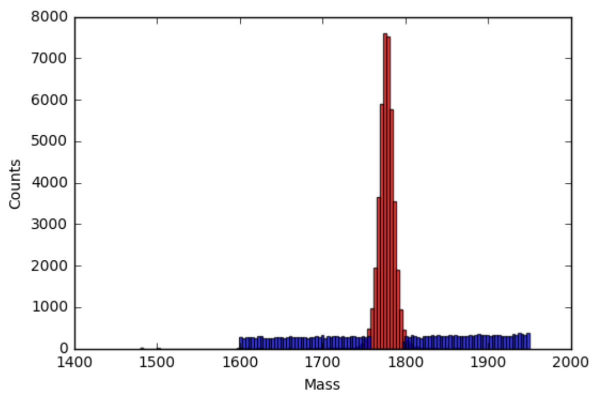

Figure 6: The gap represented here between the blue regions is an artefact of using real data mixed with simulated data in only specific regions.

4 Simulation Artifacts

As mentioned above, the amount of background data involved in such rare events is too large (in the sense that the probability distribution of the background events has too large variance) to be simulated. So real data is used instead. Now, real data may contain signal: in this case signal data may be mixed in and incorrectly labeled as background. To avoid this, the background data has been purposely chosen in a mass (or energy) range where the $\tau^- \to \mu^+\mu^-\mu^-$ cannot happen. The resulting distribution of data in terms of the mass is shown in Figure 6. We see a clear gap between the simulated signal (in red) and the real background data (in blue). This means the mass (or energy) of the event, which is also the mass of the reconstructed τ particle, can serve as perfect discriminator between signal and background. However this gap being a pure artifact of the dataset construction, a classifier that would take advantage of it would be useless for physics.

Thus an important requirement for a physically relevant classifier is that the result must not be correlated with mass (in particular this means the tau mass reconstructed feature should of course not be used). To ensure this, an additional test has been designed by the organizers[5]:

Correlation test This test measures the correlation of the model output with the mass using a Cramer-von-Mises (CVM) test. Since the model should not be relying on this feature, a low correlation score is good. The threshold for CVM is < 0.002.

More generally, we need to make sure that the classifier does not pick features that are not perfectly modeled in the simulation which would artificially boost its performance – still at the price of physical relevance. The following ‘agreement test’ checks on a specific control channel - an easily simulated event similar in mass and topology to $\tau^- \to \mu^+\mu^-\mu^-$ – that the classifier does not have a large discrepancy when applied to data and to simulation:

Agreement test This test uses the Kolmogorov-Smirnov test to compare the distributions given by he classifier on a purely simulated and a real data for the control channel. A similar distribution gives a lower score, which in turn means that the classifier is *not* exploiting simulation artifacts. The threshold for KS is < 0.09.

The results for our classifiers (with no additional feature) are shown in table 2. Naive Bayes and the single decision trees being weak classifiers, it is not surprising that they pass the two tests. We see that our boosted classifier achieves a high score without correlating with the mass (which shows in passing that the mass is manifestly a hard feature to learn). However, it fails the agreement test indicating a dependency on the simulation artifacts.

In the following section we use deep learning techniques to tentatively build a very strong classifier that would not pick up on artifacts of the simulation and be useful for physics.

[5]It seems that this test can easily be hacked, as several top ranked participants to the competition realized, see https://www.kaggle.com/vicensgaitan/flavours-of-physics/clipping-spreading

Table 2: Correlation and Agreement test results

	Naive Bayes	Decision Tree	Adaboost
AUC score	0.9613	0.9725	0.9872
Correlation Test (CvM)	0.0010 (pass)	0.0011 (pass)	0.0008 (pass)
Agreement Test (KS)	0.0307 (pass)	0.0355 (pass)	0.0907 (**fail**)

5 Deep Learning Approach

Our deep learning solution consists of a relatively standard three hidden layer neural network with the RELU activation (Glorot *et al.*, 2011). We use layer normalization on all layers (Lei Ba *et al.*, 2016). We also use dropout (Hinton *et al.*, 2012) on the first two hidden layers where we drop roughly 10% of the hidden units. This is smaller than the more conventional rate of 50%. In our first hidden layer we use 75 units, in our second hidden layer we use 50 units, and in our third hidden layer we use 25 hidden units. We implemented our solution using the Theano framework (Al-Rfou *et al.*, 2016).

The model is evaluated in terms of AUC as well as passing the agreement and correlation tests. AUC and the other tests are all defined over the entire dataset and are not well defined for a single example. However, it is known that training deep networks with current methods works best when using small batch sizes as deep neural networks trained with large batches or full gradient descent tend to have very bad generalization error (Keskar *et al.*, 2016), which cannot be solely attributed to variance reduction (Alain *et al.*, 2015). Additionally, the best results with Deep Learning have been achieved using gradient descent, which (if done exactly) requires that the loss be differentiable. Even for full gradient descent, the AUC is discrete and not differentiable, though there is some research on differentiable approximations to AUC (Herschtal and Raskutti, 2004).

Our solution to this issue is to first train a deep neural network using minibatch SGD with small batches while optimizing for cross-entropy (a standard classifier loss). Then we use a non-gradient optimization algorithm over the entire dataset to optimize for the full objective, while using the weights from the first phase as an initial solution. The specific non-gradient optimization algorithm that we use is the Powell Search Method (Powell, 1964). An interesting alternative might involve using a learned discriminator keep the training and testing characteristics aligned, rather than simply using the network resulting from the second stage (Lamb *et al.*, 2016; Ganin and Lempitsky, 2014). This could improve generalization or improve scaling to problems where the Powell Search Method cannot not be applied to the entire dataset.

We note that, because our algorithm relies on optimization over the entire dataset in the second stage, it is not scalable to datasets with a large number of examples. Because our dataset was small enough to iterate over many times (and fit into memory) we did not consider stochastic optimization for our second stage algorithm. Our second stage loss has three terms: AUC, KS-test, and CVM-test. Efficient stochastic optimization of AUC has been thoroughly studied in the literature (Gao *et al.*, 2013; Ding *et al.*, 2016; Ying *et al.*, 2016). Creating algorithms that also stochastically optimize for compliance with the KS-test and CVM-test would be an interesting area for future work.

Note that we use an ensemble of ten such neural networks, trained completely independently but with different random number seeds to achieve our final result of **0.99435 AUC** on our own validation set. This would achieve 10th place on the official Kaggle leaderboard.

Acknowledgments

We thank Pascal Vincent for his helpful feedback.

Contributions

A large bulk of the work involved in attempting this challenge was in the understanding of the domain, data, features, evaluation and the constraints imposed. **Aristide Baratin** and **Shawn Tan** took the lead in investigating the problematic of the challenge, the data and the specific difficulties arising from simulation. They are the first contributors to the present report.

Pierre-Andre Brousseau is the principal investigator and first contributor of the algorithms presented in Section 3. In particular he implemented Adaboost which overall gives excellent results despite its failing the agreement test.

Anirudh Goyal and **Alex Lamb** developed the deep learning approach described in the last section, which produced our final and best result.

References

Al-Rfou, R., Alain, G., Almahairi, A., and et al. (2016). Theano: A python framework for fast computation of mathematical expressions. *CoRR*, **abs/1605.02688**.

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. (2015). Variance Reduction in SGD by Distributed Importance Sampling. *ArXiv e-prints*.

Chrzaszcz, M., Lesiak, T., and Lusiani, A. (2015). Searches for charged lepton flavour violation. *Scholars Press*.

Ding, Y., Zhao, P., Hoi, S. C. H., and Ong, Y. (2016). Adaptive subgradient methods for online AUC maximization. *CoRR*, **abs/1602.00351**.

Ganin, Y. and Lempitsky, V. (2014). Unsupervised Domain Adaptation by Backpropagation. *ArXiv e-prints*.

Gao, W., Jin, R., Zhu, S., and Zhou, Z. (2013). One-pass AUC optimization. *CoRR*, **abs/1305.1363**.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In G. J. Gordon and D. B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings.

Herschtal, A. and Raskutti, B. (2004). Optimising area under the roc curve using gradient descent. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 49–, New York, NY, USA. ACM.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, **abs/1207.0580**.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, **abs/1609.04836**.

Lamb, A., Goyal, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29*, pages 4601–4609.

Lei Ba, J., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. *ArXiv e-prints*.

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, **7**(2), 155–162.

Ramond, P. (1999). Journeys beyond the standard model.

Rish, I. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York.

Roe, B. P., Yang, H.-J., Zhu, J., Liu, Y., Stancu, I., and McGregor, G. (2005). Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **543**(2), 577–584.

Ying, Y., Wen, L., and Lyu, S. (2016). Stochastic online auc maximization. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 451–459. Curran Associates, Inc.