Embedded Constrained Feature Construction for High-Energy Physics Data Classification

Noëlie Cherrier[1,2] **Maxime Defurne**[1] **Jean-Philippe Poli**[2] **Franck Sabatié**[1]
[1] Irfu, CEA, Université Paris-Saclay, 91191, Gif-sur-Yvette cedex, France.
[2] CEA, LIST, 91191, Gif-sur-Yvette cedex, France.

Abstract

Before any publication, data analysis of high-energy physics experiments must be validated. This validation is granted only if a perfect understanding of the data and the analysis process is demonstrated. Therefore, physicists prefer using transparent machine learning algorithms whose performances highly rely on the suitability of the provided input features. To transform the feature space, feature construction aims at automatically generating new relevant features. Whereas most of previous works in this area perform the feature construction prior to the model training, we propose here a general framework to embed a feature construction technique adapted to the constraints of high-energy physics in the induction of tree-based models. Experiments on two high-energy physics datasets confirm that a significant gain is obtained on the classification scores, while limiting the number of built features. Since the features are built to be interpretable, the whole model is transparent and readable.

1 Introduction

The entire universe is made of elementary particles such as leptons and quarks as explained by the standard model. To test this model, particle physicists try to recover properties of composite particles from these of elementary particles. The Higgs mechanism [1] and the strong interaction [2] are actively studied in particle collisions to understand the origin of mass.

The principle of a data analysis is always the same: detect and combine particles from collisions and use energy/momentum conservation to isolate the signals of interest. These signals are usually rare with several sources of background. Machine learning techniques can make a major contribution to such an analysis as long as they pass the mandatory interpretability requirement, sometimes at the expense of performances: any detail of an analysis must be explainable and understood.

Decision trees are very good candidates to meet this transparency requirement [3]. However it is also well known that the choice of features can greatly affect their performances [4]. Generally, a preprocessing step of feature engineering is performed manually based on domain expertise. In particle physics, quantities related to energy/mass/momentum balances and highly dependent on the process of interest are derived from base variables. Although understandable and analyzable by construction, nothing guarantees that these quantities are optimized for the analysis of the process of interest.

The field of feature construction (FC) aims at automating the feature engineering step. In particular, embedded FC permits to better adjust the built features to a local data discrimination problem during the model induction. In this work, we aim at developing techniques of embedded FC with a special attention to the readability of the final model. The global transparency of a model also requires the interpretability of the built features themselves, which is the second focus of our work.

We first review related work in the field, before presenting our method and conducting experiments on two HEP datasets from two different physics studies: the Higgs mechanism at CERN on one hand and the strong interaction at Jefferson Laboratory on the other hand.

2 Related work

Most of the automatic feature generation algorithms are actually preprocessing methods: new features are built from mathematical functions of the original ones and the training is performed in a subsequent step with the new features.

The literature in automatic FC is very abundant and a survey can be found in [5]. Genetic programming (GP) algorithm is a popular method in the FC field [6, 7, 8, 9]. It consists in evolving a population of N-tree-like individuals through mutations and crossovers while selecting good individuals for the next generation.

In [10], the authors propose a method to adapt a GP algorithm to handle the constraints of HEP. Indeed, in HEP, a feature is interpretable if it resembles some of the physics laws ruling our universe. These physics laws connect variables carrying compatible units: for instance, no physics formula exhibits the raw sum of an angle and an energy. Therefore, to enforce the combination of compatible features, Cherrier et al. [10] constrain the GP algorithm with a grammar and a transition matrix. The resulting features are interpretable to physics experts and the classification score is improved compared to the unconstrained version of the algorithm.

On the other hand, the field of embedded FC has not been so much explored yet. Ekárt and Márkus [11] are the first to use a GP algorithm to find the best splitting feature at each node in the induction of a decision tree. Maes et al. [12] embed a Monte Carlo search of features in several tree-based ensemble methods, building one feature at each node of the tree.

3 Embedded constrained feature construction method during tree induction

Cherrier et al. [10] mainly use a machine learning algorithm to evaluate the candidate features during the evolution. In this work, we propose to use the split criterion in the decision trees or ensemble methods as the fitness function in the constrained GP algorithm of Cherrier et al. [10], thus improving the speed of the algorithm. Computing a single information gain is indeed faster than training a whole decision tree.

Besides, we experimentally limit the number of built features per tree to support overall intelligibility of the model and also to prevent overfitting. When the number of allowed constructions is restrained, the features are built from the root and level by level, going down as the tree is formed.

We propose hereafter a generic framework for embedding FC into classical tree induction algorithms. The induction of the most commonly used tree classifiers (e.g. C4.5 [13] and CART [14]) is made by sequentially constructing their nodes. For each node, the problem is to separate the data into two subsets while optimizing a criterion which depends on the particular induction algorithm. Most algorithms perform a search among the existing features to find the best split according to this criterion. We modify this feature search step by the function `findBestSplit` described in Algorithm 1.

At each node during the induction of a decision tree, if the `constructionCondition` in Equation (1) below is met with depth d and number of built features so far n_f, with a maximal allowed number N_{max} of feature constructions, we replace the classical search among existing features by a FC method.

$$d \leq \log_2(1 + N_{max}) \quad \text{and} \quad n_f < N_{max}. \tag{1}$$

This condition is designed to ensure that the feature constructions will be performed in the first layers of the tree. The final feature obtained by the FC algorithm is used as the feature to split the data at the current node.

We use the C4.5 [13], adaptive boosting [15] and gradient boosting [16] algorithms in our experiments. The induction of a C4.5 decision tree is made by maximizing the information gain at each node. Adaptive boosting with decision trees only changes the weights of training examples. However, the gradient boosting classifier uses the mean squared error (MSE) as the splitting criterion since the global classification problem becomes a regression problem in the weak tree classifiers. Feature

Algorithm 1: Generic induction of a node of a decision tree with embedded FC

```
Input : $data$ in the current node
        $n_f$ the number of built features so far in the tree
        $depth$ the depth of the current node
        $N_{max}$ the maximum number of features to build in the tree
Result: the inducted node
Function findBestSplit(data, depth, $n_f$)
    if constructionCondition ($n_f$, depth) then
        build splittingFeature with splittingCriterion as fitness function
        $n_f \leftarrow n_f + 1$
    else
        foreach feature in data do
            compute splittingCriterion on feature
        splittingFeature $\leftarrow$ feature obtaining the best criterion
    if splittingFeature does not satisfy specific requirements then return null
    splitted $\leftarrow$ split data along splittingFeature
    return splitted
```

construction is done by replacing the fitness function by the splitting criterion of the induction algorithm, computed using the considered candidate feature.

4 Experiments

We consider two HEP classification problems in this study.

DVCS dataset At Jefferson Laboratory, an electron beam scatters off protons at rest in the lab frame. The objective is to discriminate between the DVCS interaction whose final state is composed of an electron, a proton, and a photon, and the π^0 production event which has a similar final state, except that the photon is replaced by a π^0. The later immediately decays into two photons and one of them may not be detected, mimicking a DVCS event. The available features are the three-dimensional momentum for each identified particle.

Higgs dataset [17] At CERN, two protons collide head-on with each other and Higgs particles are notably produced out of the collisions. The objective of this dataset is to detect Higgs bosons decaying into two τ-particles. The simulated data is publicly available on the Open Data platform of CERN. 17 primitive features per event are available, including notably several geometrical features for each detected particle.

For both datasets, we use 100000 instances: 80% of the them are dedicated to training and 20% to performance evaluation. One should note that both datasets come from simulated data since the truth information is needed for training. Features already present in the datasets include the three-dimensional momenta of the particles as well as their θ and ϕ angles. For the GP algorithm, we evolve a population of 500 individuals during 70 generations. For ensemble methods, we use a downgraded version of GP called "GP down" performing only 6 generations.

4.1 Performance evaluation

In this paragraph, we evaluate the performances of C4.5, AdaBoost and GradientBoosting classifiers with the embedded GP algorithm described above.

We vary the number of features built in total for C4.5 or per tree for ensemble methods. We use the Cohen's kappa metric to evaluate the performances, with at least 10 independent runs per displayed value. Figure 1 displays the evolution of the Cohen's kappa score of these three tree-based models depending on the number of built features. For ensemble methods, we reserve the possibility to build less than one feature per tree in the ensemble: in the x-axis on the top of the graph, a value p below 1 means that each tree in the ensemble has probability p to build one feature. Above 1, the value p can only be an integer and is the number of features built per tree. The expectation for the total number

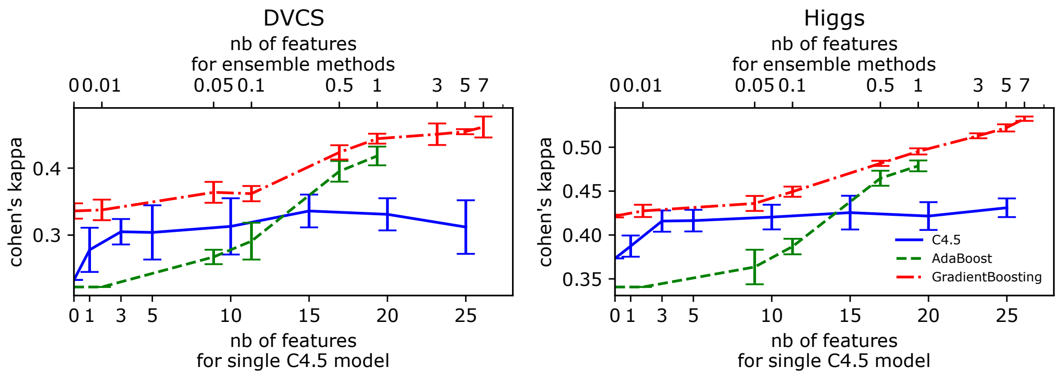

Figure 1: Impact of the number of features on the classification score. The scale on the y-axis is different from one graph to the next.

Table 1: Performance comparison (Cohen's kappa metric).
[*] For DVCS, two hidden layers with respectively 100 and 50 neurons. For Higgs, one hidden layer with 150 neurons. This is the configuration leading to the best score after a grid search on up to three hidden layers and from 50 to 150 neurons per layer.

	DVCS	Higgs
C4.5	0.223	0.373
AdaBoost	0.223	0.341
GradientBoosting	0.336 ± 0.011	0.4216 ± 0.0016
Neural network[*]	0.333 ± 0.036	0.337 ± 0.062
Linear SVM	0.343	0.169
FC C4.5 best	0.3446 ± 0.0080	0.450 ± 0.013
FC AdaBoost best	0.424 ± 0.014	0.4787 ± 0.0062
FC GradientBoosting best	$\mathbf{0.461 \pm 0.016}$	$\mathbf{0.5325 \pm 0.0025}$

of built features in the whole algorithm is then the product of this value p with the number of trees in the ensemble.

The results show that the score increases with the number of built features per tree for ensemble methods. However, it stagnates for C4.5 algorithm after around 10-15 built features. This shows that embedded FC with constrained GP brings a significant gain to the classification score compared to using the same tree-based algorithm without FC (score corresponding to zero built feature on Figure 1. Moreover, a reduction of the number of built features in C4.5 can be performed without impairing the score.

Table 1 provides a few baselines, including the three tree-based algorithms without FC, a neural network and a SVM. These baselines are compared to the best scores (i.e. the highest point on Figure 1) obtained with the three tree-based algorithms with embedded FC. Even compared to black-box models which are supposed to have an internal complex representation of the feature space, the three presented algorithms with embedded FC perform better.

4.2 Model readability

Model conciseness is supported by the smaller number of built features required by C4.5 to get an optimal classification score, and by the small size of weak tree classifiers which are limited in depth in the used implementations of the two presented ensemble methods (depth 1 for AdaBoost and 3 for GradientBoosting). The final feature space obtained with embedded FC is hence of reduced size.

Equations (2) and (3) display two features built for the Higgs dataset that are either recurrent in their simple form or recurrent as a pattern in more complex features (they appear in respectively 61% and 74% of runs):

$$\cos\left(\theta^{lep} - \theta^{\tau}\right) \qquad (2) \qquad\qquad \cos\left(\phi^{lep} - \phi^{\tau}\right) \qquad (3)$$

These features visibly compares the geometrical angles of a lepton and a τ, two particles which are indeed expected to be the products of the decay of the same particle. The mass of that particle can be computed from the features displayed above.

Equation (4) shows a recurrent feature built for the DVCS dataset (it appears in 79% of runs):

$$p_z^e + p_z^{\gamma_1} + p_z^p \tag{4}$$

This feature is a momentum conservation check along the beam direction, absolutely relevant considering the detector and event geometries of a fixed-target experiment. A π^0 event would obviously miss a second photon momentum in the p_z sum so this feature would not take the same value.

5 Conclusion

In the machine learning field, there is generally a need to strike a balance between the interpretability and the performances of a model. In this work, without loss of generality, we improved three tree-based models by embedding a constrained FC technique adapted to physics problems during the induction. In any case, embedded FC permitted to improve the classification score while experimenting on two datasets of HEP. The constrained versions of the FC methods enables to build features that are interpretable at least to the experts of the field.

With the proposed method, the final model is more performant while remaining readable for further interpretation. Finally, instead of choosing between performance and interpretability, we increased performance while focusing on keeping the readability of the final model.

References

[1] Peter W. Higgs. Broken symmetries and the masses of gauge bosons. *Phys. Rev. Lett.*, 13: 508–509, Oct 1964.

[2] Craig D. Roberts. Perspective on the origin of hadron masses. *Few Body Syst.*, 58(1):5, 2017.

[3] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning. *arXiv:1806.00069 [cs, stat]*, May 2018.

[4] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant Features and the Subset Selection Problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 1994.

[5] Parikshit Sondhi. Feature Construction Methods: A Survey. page 8, 2009.

[6] Krzysztof Krawiec. Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery Tasks. page 15, 2002.

[7] Fernando E. B. Otero, Monique M. S. Silva, Alex A. Freitas, and Julio C. Nievola. Genetic Programming for Attribute Construction in Data Mining. In *Proceedings of the 6th European Conference on Genetic Programming*, EuroGP'03, pages 384–393, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 978-3-540-00971-9.

[8] Matthew G. Smith and Larry Bull. Genetic Programming with a Genetic Algorithm for Feature Construction and Selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, September 2005.

[9] Kourosh Neshatian and Mengjie Zhang. Genetic Programming and Class-Wise Orthogonal Transformation for Dimension Reduction in Classification Problems. In *Genetic Programming*, Lecture Notes in Computer Science, pages 242–253. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78671-9.

[10] N. Cherrier, J. Poli, M. Defurne, and F. Sabatié. Consistent Feature Construction with Constrained Genetic Programming for Experimental Physics. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1650–1658, June 2019.

[11] Anikó Ekárt and András Márkus. Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms. *AI EDAM*, 17(03), August 2003.

[12] Francis Maes, Pierre Geurts, and Louis Wehenkel. Embedding Monte Carlo Search of Features in Tree-Based Ensemble Methods. In *Machine Learning and Knowledge Discovery in Databases*, volume 7523, pages 191–206. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[13] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[14] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[15] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. *Statistics and its interface*, 2, 02 2006.

[16] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232, 10 2001.

[17] Claire Adam-Bourdarios, Glen Cowan, Cecile Germain, Isabelle Guyon, Balazs Kegl, and David Rousseau. Learning to discover: the Higgs boson machine learning challenge - Documentation. 2014.