Embedded Feature Construction in Fuzzy Decision Tree Induction for High Energy Physics Classification

Noelie Cherrier, Jean-Philippe Poli, Maxime Defurne, Franck Sabatie

Embedded Feature Construction in Fuzzy Decision Tree Induction for High Energy Physics Classification

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

April 22, 2024

Abstract

Fuzzy decision trees have been successfully applied in numerous domains. The popularity of these models comes notably from their interpretability, namely the ability of humans to understand them. However, on the contrary to neural networks, the induction of such models does not include a generation of their own feature space. In this work, the embedding of feature construction in fuzzy decision tree induction algorithms is studied, so that they can create new input features, without affecting the overall interpretability of the model. This method is successfully applied to a classification problem in high-energy physics to study the benefits of having constructed features in fuzzy decision tree on the classification scores, allowing them to have their own interpretable representation of the data.

Keywords: Fuzzy decision trees, genetic programming, feature construction, interpretability.

1 Introduction

Decision trees have been applied to many problems since the early works that allowed automatic induction [1, 2]. Today, they are still studied despite the hegemony of deep neural networks that lead to unprecedented classification scores. However, for some applications, it is necessary to be able to interpret the model, and even to provide an explanation [3]. One can think of human-centered applications, such as medicine and security [4], but it is also true for scientific applications, such as physics. In particular, this desire is met in High Energy Physics (HEP), which aims at studying the elementary constituents of matter and the fundamental forces that govern them. It is crucial for the

fundamental sciences to be able to interpret automatically generated models and compare them with standard analysis methods for validation.

Neural networks do not offer this transparency, hence a renewed interest for decision trees, even if "boosted" versions or forests are also hardly interpretable. However, decision trees have several drawbacks with respect to neural networks: crisp decision boundaries parallel to the axes, absence of an automatic data representation, etc.

Fuzzy decision trees [5–7] correct the boundary issues but do not allow a better representation of the data. This work addresses this problem. It is important to create a representation of the data that remains interpretable so as not to affect the quality of the fuzzy decision tree. The field of feature construction aims at automating the feature engineering step. In particular, embedded feature construction permits to better adjust the built features to a local data discrimination problem during the model induction. We therefore propose to automatically create features during the induction of the tree and test our approach on a dataset from a HEP experiment.

Indeed, HEP, as other experimental sciences, has everything to gain from interpretable models which can provide new avenues of research after studying the model and its inference process.

The article is composed as follows: the following section presents the previous work about fuzzy decision trees and feature construction. The third section describes our work on integrating the feature construction within the C4.5 algorithm and two fuzzy versions of this algorithm. The next section describes the experiments carried out to validate our approach, showing a gain in classification performance on our dataset. Finally, we draw conclusions and perspectives for this work.

2 Background

2.1 Fuzzy Decision Trees

Decisions trees are efficient models to classify data, which became popular with the apparition of induction algorithms like *Classification And Regression Trees* (CART [1]), *Iterative Dichotomiser 3* (ID3 [2]) and C4.5 [8]. They are particularly appreciated for both their interpretability and their efficiency. Fuzzy decision trees [9] have then been introduced, taking advantage of fuzzy logic to deal with uncertainty of knowledge and the possibility to belong to several classes at the same time [5].

Most of the techniques to build fuzzy decision trees rely on a top-down approach [7]:

1. an attribute A is selected regarding a measure of discrimination H [10]
2. according to this attribute A, the dataset is split into several subsets
3. if a subset meets a stop criterion, a leaf is created, otherwise, the algorithm resumes from the first step.

The different algorithms differ from each other regarding the function H that they use, for instance the entropy star measure [11], and the way they create modalities for each variable [6, 12].

In particular, some techniques create modalities by fuzzifying the thresholds involved in the nodes [13, 14]. Thus, instead of sharply dividing the data into two distinct branches like crisp decision trees, the separation boundary is now a fuzzy transition between the left and right child nodes. For instance, in the fuzzy version of the SLIQ algorithm [13], the separation function of the two branches depends on the standard deviation of the attribute. In [14], the authors do a Fibonacci search to optimize the slope of a linear transition, after having determined the optimal threshold with a CART-like algorithm.

To the best of our knowledge, even if we can find a lot of works concerning the creation of modalities (i.e. terms of linguistic variables involved in the decision process) or the fuzzification of the thresholds, the construction of new attributes from original attributes themselves has not been studied in fuzzy decision tree induction. In the remainder of the article, we will use indifferently the terms "feature" and "attribute" (that designate the same concepts in the different communities, respectively feature construction and fuzzy decision tree induction).

2.2 Automatic Feature Construction

This section focuses on the explicit construction of new features. We assume that we have a set of initial features and a list of construction functions applicable to these features. These functions are related to the domain of application.

Feature construction algorithms usually split into two categories according to the criterion used to evaluate the candidate features [15, 16]: the "filter" methods, which use statistical criteria such as the gain of information or the Fisher criterion [17] on the one hand; and "wrapper" methods that use the score obtained by a predictive model driven with the candidate feature [18] on the other hand. The literature in the field of feature construction is very abundant: a more complete review can be found in [19] or [16]. Evolutionary methods represent the vast majority of the proposed methods, including genetic programming [18, 20–22]. There are also papers using particle swarm optimization algorithms [23] or grammatical evolution [24]. Besides evolutionary algorithms, tree-based search algorithms are also used in some works, for example FICUS [25] and Cognito [26].

Most of the automatic feature construction algorithms are considered as preprocessing: they allow to extract new features that are mathematical functions of the original ones. Machine learning models are then trained with these new features. However, certain methods allow building features during the training algorithms. These methods mainly use simple decision trees; for instance, Yang et al. [27] propose a scheme to simplify parts of a decision tree with recurrent boolean patterns, Ittner and Schlosser [28] exhaustively build all the additions and products of the original features, or Argyriou et al. [29] extract linear features by solving a convex optimization problem. We can also cite Ekárt

and Márkus [30], who are the first to use a genetic programming algorithm to find the best feature on which to divide the current node. Finally, Maes et al. [31] incorporate a Monte Carlo feature search into several tree-based induction methods, constructing one feature at each node of the tree. In [32] and further in [33], the authors use genetic algorithms both for FC and model induction. Related to this area is the domain of oblique decision trees [34]. The principle is to find the optimal splitting variable as a linear combination of the input features. Multivariate decision trees have also been developed [35]. However, Yildiz and Alpaydin [35] state that complex models of the input features may lead to overfitting especially when used in the deep layers of the tree.

In [36], the authors propose an adaptation of the genetic programming algorithm for feature construction that allows producing interpretable features, in particular by respecting constraints related to the nature of the original features. For example, in high energy physics, a feature is considered interpretable by domain experts if it relates to some of the physical laws that govern the universe. These physical laws combine variables while respecting their dimensions: for example, no physical formula will sum an angle and an energy. Thus, the method proposed in [36] constrains the feature construction with a grammar and a matrix of transition probabilities between operators to force the combination of compatible original features only. The features created by this method are interpretable for the physics experts and allow improving the classification score compared to the unconstrained version of the algorithm.

This method builds new features first and then drives an automatic learning algorithm with them. The next section deals with the integration of the feature construction during the induction of fuzzy decision trees.

3 Embedding feature construction in fuzzy decision tree induction

The method proposed in [36] builds new features upstream of the induction algorithm. In this case, it would be enough to replace the fitness function used in [36] by the score obtained by the fuzzy decision tree induced by one of the algorithms presented in the previous section, using the newly built features. This results in a set of features specifically designed to obtain a good classification score with this algorithm.

However, it may be wiser to integrate the feature construction process during the creation of the nodes themselves. Indeed, each node of the tree must solve a more specific problem than its parent, having a different distribution of the data set. The feature optimizing the measure of discrimination H at this point can therefore differ, hence the idea of using the feature construction algorithm directly when choosing the feature in a node.

We first describe the construction of consistent features and we then describe the modified fuzzy decision tree induction algorithm.

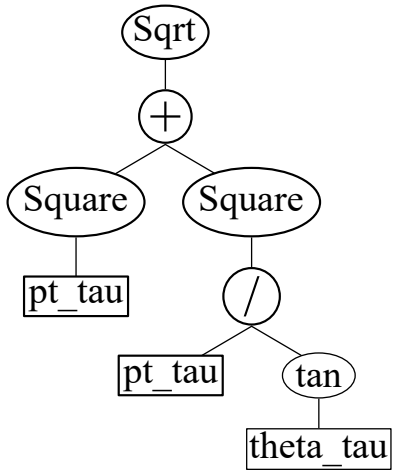


Figure 1: Example of a candidate feature represented by a tree: $\sqrt{{p_T^\tau}^2 + \left(\frac{p_T^\tau}{\tan\theta^\tau}\right)^2}$, which could be a real feature constructed by a constrained FC algorithm on the DVCS dataset further presented.

3.1 Constrained Genetic Programming for Feature Construction

The genetic programming consists here in evolving a population of N_{pop} tree-like individuals through mutations and crossovers while selecting good individuals for the next generation. An individual in this algorithm is an expression tree representing a candidate feature, such as the one displayed on Figure 1. Any internal node is a mathematical operator and leaves are original attributes. Such algorithm is free to combine any base feature, making it less adapted regarding any interpretability requirement.

Indeed, in sciences, and in particular in physics, variables are connected according to their units. Therefore, to enforce the combination of compatible attributes, we constrain the genetic programming algorithm with a grammar and a transition matrix as described in [36]. The resulting features are interpretable to physics experts and the classification score is improved compared to the unconstrained version of the algorithm.

To summarize, the steps of the algorithm are as follows:

1. Initialization: N_{pop} trees are generated randomly using the ramped half-and-half initialization [37], but still respecting the grammar and the transition matrix while choosing the operators.

2. Evolution during K generations:

   (a) Perform mutation and crossover: the methods are inspired from

generic techniques in genetic programming [37], but altered to ensure the children trees still represent valid features with respect to the grammar and transition matrix.

(b) Evaluation: the fitness function of a candidate feature becomes here the optimal discrimination measure H computed regarding the modalities of the new feature.

(c) Selection: a repeated tournament selection is performed among three individuals each time to form the next generation.

3. End of the algorithm: return the individual with the highest fitness.

3.2 Modified Fuzzy Decision Tree Induction

We therefore modify the general fuzzy decision tree induction algorithm presented in section 2.1.

To maintain the readability of the tree, a parameter controls the maximum number N_{max} of features that it is authorized to build.

The general algorithm for fuzzy decision tree induction with embedded feature construction is presented in Algorithm 1.

For instance, the *constructionCondition* can be the condition in Equation (1) below: if it is satisfied for a depth d in the tree and a number of features built so far n, then the algorithm does not select an original attribute regarding the H function but creates instead a new feature with the algorithm presented in section 3.1. This feature construction algorithm can actually build features represented by trees of depth 1, which are the "rebuilt" raw features. These raw features might then have been feature candidates but discarded during the construction process.

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

4 Experiments

This section analyzes the benefits of embedded feature construction to the scores of fuzzy decision trees. The importance of the number of constructed features in the tree is also studied. Next subsection presents the application.

4.1 Classification of CLAS12 Data

At Jefferson Laboratory, an electron beam scatters off protons. The objective is to discriminate between two types of events: on the one hand, the γ^{DVCS}-events, whose final state is composed of an electron, a proton and a photon noted γ, and on the other hand the $\gamma\gamma^{\pi^0}$-events which have a similar final state, except that two correlated γ photons are produced. The three-dimensional momentum (i.e. mass times speed) are available for each identified particle, for a total of 15 basic features with no missing values. The training of the algorithms is

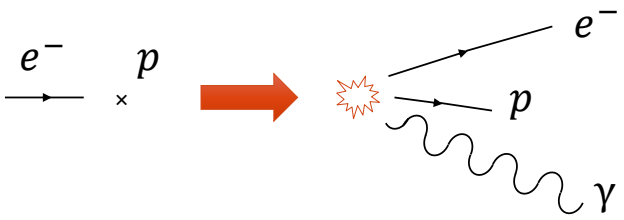
Algorithm 1: Generic induction of a decision tree with embedded feature construction.

Input: *data*, subset of the dataset
n, number of built features so far in the tree
depth, depth of the current node
N_{max}, maximum number of features to build in the tree

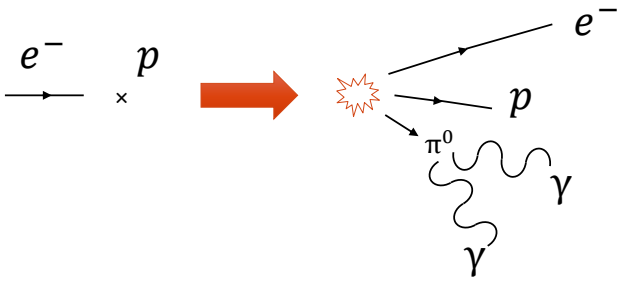
Result: a tree and the number of constructed feature

Function `createFDT(`*data, n, depth,* N_{max}`)`

- $r \leftarrow$ create a new node
- **if** `constructionCondition` *(n, depth)* **then**
  - $A \leftarrow$ build new feature
  - $n \leftarrow n + 1$
- **else**
  - $A \leftarrow$ select the best original attribute regarding the discrimination measure H
- Affect to r a test on A
- $M \leftarrow$ create a set of modalities $m_1, \ldots, m_t$ for A
- Split data into $d_1, \ldots, d_t$ regarding M
- **foreach** *couple* (m_i, d_i) **do**
  - **if** *stoppingCriterion*(m_i, d_i) **then**
    - create a branch m_i from r to a leaf built with statistics of d_i
  - **else**
    - create a branch m_i from r to `createFDT` (d_i, n, depth+1, N_{max})
- **return** (r, n)



(a) DVCS interaction



(b) π^0 interaction

Figure 2: Schemas of two possible reactions.

done on simulation data. About 15000 events are used here, half of which are γ^{DVCS}-events and the other half are $\gamma\gamma^{\pi^0}$-events.

Figure 2 presents the two kinds of events. It shows that the π^0 leads to two photons whereas the DVCS event leads to only one. The difficulty lays in the fact that photons are not always detected.

Classically, physicists build mathematical functions of these input variables, for example energy or momentum balances, or comparisons of theoretical and experimental angles. The momenta themselves provide little information on the nature of the reaction that took place, hence the need to construct new variables that are more relevant.

4.2 Implementations

Without loss of generality, we had to choose a few fuzzy decision tree induction algorithms to perform our experiments.

For tests convenience (availability of source code, efficiency of implementations), we used modified versions of existing fuzzification of C4.5 algorithm [38]. It builds nodes sequentially from the root to the leaves. In this case, the dis-

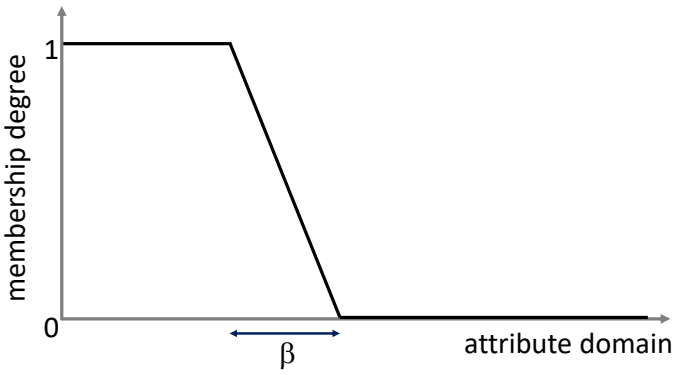


Figure 3: Illustration of the parameter β [14].

crimination measure H is the information gain. At each node, the problem is to divide the data into two subsets in order to maximize information gain. Given a parent node with a data set X_0 and two children data nodes X_1 and X_2, the information gain for the parent node is written as follows:

$$IG = E(X_0) - E(X_1) - E(X_2) \tag{2}$$

$$E(X) = -\sum_k p_k(X) \log_2 p_k(X) \tag{3}$$

with $E(X)$ the entropy of a set X and $p_k(X)$ the frequency of the class k in the set X.

In the fuzzy version of the induction algorithm, the separation of data in two is not crisp. Each example x is actually assigned a membership degree $d_{n_i}(x)$ to each of the children nodes n_i. The frequency in the computation of entropy of Equation (3) is thus replaced by a sum weighted by the membership degrees:

$$IG = E(X, n_0) - E(X, n_1) - E(X, n_2). \tag{4}$$

$$E(X, n) = -\sum_k f_{k/n} \log_2 f_{k/n}. \tag{5}$$

$$f_{k/n} = \frac{\sum_{x \in k} d_n(x)}{\sum_x d_n(x)}. \tag{6}$$

The modalities are then computed by fuzzifying the thresholds obtained with the C4.5 algorithm. For the sake of simplicity and efficiency, only piecewise linear membership functions are considered, with a slope parameterized by a real value β representing its width, as show in Figure 3.

Two different algorithms can determine the parameter β. The first is inspired by [13]: the shape of the membership function depends on the standard

deviation σ of the values that has been split, in a proportional way such that all that remains is to set a free parameter α:

$$\beta = \alpha\sigma. \tag{7}$$

The second algorithm was proposed in [14], which makes a Fibonacci search of the value of β in order to optimize the criterion already used to choose the feature and the threshold. This is a technique based on the principle of divide and conquer by exploiting Fibonacci numbers. It assumes that the function to be optimized is unimodal, that is, it is strictly increasing (or decreasing) up to the maximum (or minimum) value of the function, then strictly decreasing (or increasing). In practice, the number of evaluations with one parameter N is limited in the algorithm. On the other hand, Olaru et al. show [14] that looking for the width β only after setting the threshold does not deteriorate the criterion that one seeks to optimize.

4.3 Experimental Protocole

We divide the data presented above into 80% for learning and 20% for performance evaluation, with balanced classes. In these experiments, we did not perform any cross validation for several reasons: the simulated data are numerous enough to be sure of the representativeness of the learning and validation data sets. Moreover, since the feature construction is stochastic, the tree induction is run several times with the same inputs to measure the stability of the embedded feature construction, and it is obviously time consuming.

The adapted C4.5 algorithm uses the same parameters as recommended in [8], namely a minimum number of instances per leaf equal to 2. However, no pruning step is performed, to reduce the computation time and because the goal of our study is to prove the contribution of embedded feature construction to a specific learning algorithm, not to obtain the best possible results for each data set.

For the "Fuzzy-std" version of C4.5, which uses the standard deviation of the data, α is set to 0.1. For the "Fuzzy-fibo" version that uses a Fibonacci search, the maximum number of evaluations of the parameter β is set to $N = 5$ just as in [14].

The performance criterion used below to evaluate the different algorithms is the classification accuracy, since the two classes used in our dataset are present in equal quantities. Mean and standard deviation of accuracy are systematically presented on at least 10 independent experiments.

4.4 Results

Table 1 compares the classification accuracies obtained by the simple C4.5 algorithms, the Fuzzy-std version (F-std in the table), and the Fuzzy-fibo version (F-fibo in the table) by building from 0 to 100 features. The scores in bold show the best method according to N_{max}. For each score, the mean value and the

standard deviation are shown. The choice of maximum 100 features is explained by the observation that the generated trees have in average up to 100 nodes: allowing 100 constructed features thus means creating a new feature at each node (and thus not using the original attributes anymore in the nodes). The actual number of built features can then be smaller if the induction stops before building N_{max} nodes.

Table 1: Classification accuracies (in %) obtained by the non-Fuzzy C4.5, Fuzzy-std and Fuzzy-fibo algorithms according to the number of built features.

N_{max}	0	1	3
C4.5	67.29 ± 0.54	68.7 ± 1.5	71.14 ± 0.85
F-std	**69.60 ± 0.32**	**70.6 ± 1.2**	**72.20 ± 0.48**
F-fibo	69.10 ± 0.32	**70.5 ± 1.7**	71.42 ± 0.43
N_{max}	5	10	15
C4.5	70.6 ± 1.0	71.95 ± 0.89	72.11 ± 0.51
F-std	**72.40 ± 0.91**	72.85 ± 0.51	**73.32 ± 0.62**
F-fibo	71.56 ± 0.62	**73.12 ± 0.61**	**73.37 ± 0.80**
N_{max}	20	25	100
C4.5	72.48 ± 0.56	72.28 ± 0.90	71.28 ± 0.73
F-std	73.31 ± 0.64	73.31 ± 0.64	73.20 ± 0.74
F-fibo	**73.46 ± 0.90**	**73.40 ±0.63**	**73.49 ± 0.51**

Table I shows the dominance of fuzzy algorithms compared to the classic decision trees, with or without feature construction. An improved score of 0.3 to 2.3% is obtained with a fuzzy version compared to the crisp version.

There is a clear improvement in the classification score with the number of built features, for the three models. On average, the accuracy is increased by almost up to 4% by building features in tree nodes instead of using only the original features. It is interesting to note that the Fuzzy-fibo algorithm seems to benefit more from the feature construction than the Fuzzy-std algorithm. In fact, the Fuzzy-std model obtains better performances when one builds a small number of features, whereas the Fuzzy-fibo model exceeds it beyond 10 constructed features.

The score obtained by building features seems to reach a maximum, stagnating or even decreasing after a certain threshold of constructed features. Indeed, the classification accuracy of the crisp C4.5 decision trees no longer improves after about twenty built features, and even decreases afterwards. Similarly, for Fuzzy-std, a plateau is reached around 73.32% for 15 built features. However, one cannot say that this score decreases significantly when the number of features increases. The same phenomenon is observed for Fuzzy-fibo whose score does not increase significantly from 15 features.

Finally, Figure 4 depicts the evolution of the number of leaves in the tree (i.e. number of rules when converted into a rule base) in function of the number of built features. No correlation can be inferred between the size of the tree and the

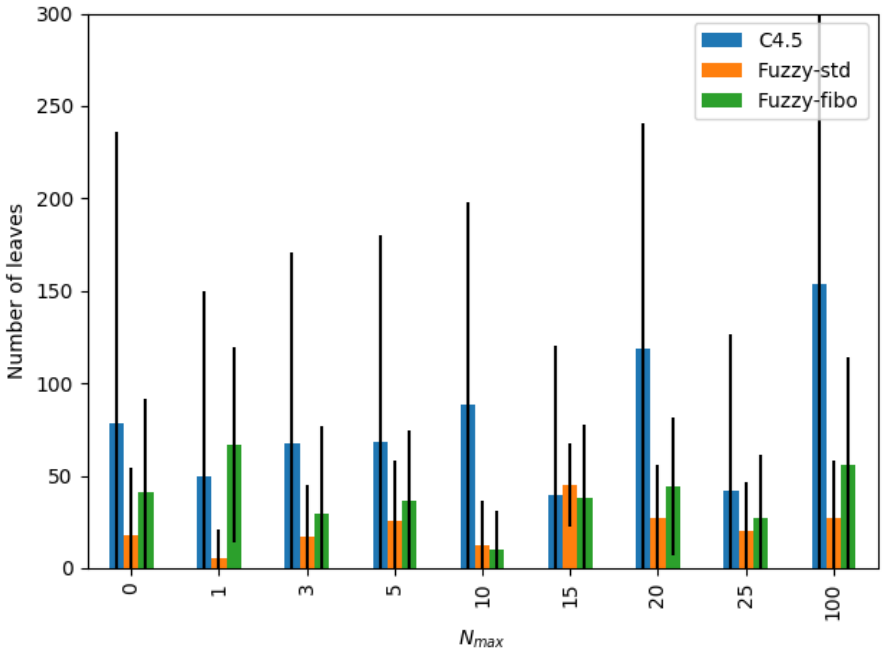


Figure 4: Number of leaves of trees generated by C4.5, Fuzzy-std and Fuzzy-fibo as a function of N_{max}.

number of built features. However, Fuzzy-std and Fuzzy-fibo are significantly smaller than crisp C4.5: the p-value of a Welch's t-test between Fuzzy-fibo and C4.5 is 0.0046 while the p-value of the same test between Fuzzy-std and C4.5 is 4.5×10^{-5}.

4.5 Interpretability

In this work, overall model interpretability relies on the inherent interpretability of the base model (decision tree) on the one hand, and on the interpretability of the attributes used at each node on the other hand, especially if these attributes are high-level combinations of the raw ones, which is the case here. The goal here is to assess the interpretability of the trained models from the point of view of experts in the field. Expert physicists are able to interpret high-level built features (physical formulas).

In 79% of the runs of the algorithm, the same feature is built: it is the sum of the z component of the momentum (i.e. mass times speed) of three particles resulting from a signal γ^{DVCS}-event, that can be written:

$$p_z^e + p_z^{\gamma_1} + p_z^p. \tag{8}$$

According to momentum conservation, the total momentum of the output particles must be equal to the momentum of the input particles, here 10.6 GeV. If the sum of the momenta of the output particles is inferior to 10.6 GeV, additional particles have been produced from the collision and hence it is not a signal event. Therefore, it is logical that this feature is relevant from the point of view of physicists as well as for decision trees.

Another common feature consists of the angle between γ_2 the lowest energetic photon and the sum of two detected photons $\gamma_1 + \gamma_2$:

$$\angle(p_{\gamma_2}, p_{\gamma_1} + p_{\gamma_2}). \tag{9}$$

A signal γ^{DVCS}-event involves a single γ photon. However, an uncorrelated photon from background may be simultaneously detected: in this case, it resembles the major background being $\gamma\gamma^{\pi^0}$-events. The two γ photons of a $\gamma\gamma^{\pi^0}$-event are correlated since produced by the decay of a same particle. Therefore, this angle is indeed a discriminative feature since it studies the correlation between the two γ photons to isolate background events.

4.6 Discussion

Overall, feature construction improves the classification score on CLAS12 data. However, a plateau is observed or even a performance degradation for the C4.5 algorithm when a certain number of built features is exceeded. This can be explained by over-fitting especially in the deepest nodes of the trees, in which the amount of data is low. Building new features in these nodes seems useless and even counterproductive.

The reason for the difference in scores between Fuzzy-std and Fuzzy-fibo remains difficult to explain. Using the standard deviation of features may allow a better generalization, while the Fibonacci search suffers from over-fitting. However, some of the built features can have a rather complex distribution, which would disadvantage Fuzzy-std compared to Fuzzy-fibo. The latter can indeed more easily optimize the width of the slope representing the fuzzy threshold. Thus, the more features are built, the better Fuzzy-fibo can perform compared to Fuzzy-std.

We chose not to fuzzify the threshold when evaluating candidate features during feature construction. It is possible that the features found are therefore not optimal for a given node. However, the computation time is shortened, which is a weighty argument when one must carry out a large number of evaluations of candidate features during the evolution of a population, which is the case in the used genetic programming algorithm. Moreover, obtaining the optimal feature is not necessary and could lead to over-fitting, as pointed out in several works [31, 35].

On the computational overhead, here is a worst-case approximation. The experiments were conducted on several multicores machines (between 12 and 24 cores) with Intel CPUs (2.20 to 2.80 GHz). One feature construction takes around thirty minutes, the longest part being the evaluation of the feature

candidates. This approximation is valid only at the root nodes as they use the entire dataset. In the deeper nodes, the local data subsets are smaller and then the feature search phase is shorter. The induction of the rest of the tree is negligible when no feature is constructed. So for instance, building ten feature in a fuzzy decision tree takes below five hours. The gain compared to using feature construction as a preprocessing step comes mostly from the feature fitness evaluation time: a single information gain is a lot faster to compute than the induction of a whole decision tree. To give an order of magnitude, building only one feature prior to model induction (hence inducing a model each time a candidate feature is evaluated, which is done in [36]) takes about 24 hours on the same machines.

Finally, the overall interpretability of the model can be discussed. The features themselves can be interpreted by specialists in the field thanks to the feature construction algorithm used [36]. On the other hand, the numerical results show that it is not necessary to build a feature at each node of the tree. About fifteen or twenty features are sufficient to achieve an optimal classification score. The feature space is therefore limited, which helps the overall readability of the model.

5 Conclusion and perspectives

In machine learning, there usually exists a balance between performance and interpretability of a model. In this work, we study the contribution of embedded feature construction on the performances of fuzzy decision trees, while limiting the overall dimensionality of the model, and producing only interpretable features. The goal is to allow fuzzy decision tree induction algorithms to create a more suitable data representation by creating new features that are correct mathematical formulas of the original attributes.

This work proposes to integrate an algorithm for constructing interpretable features in the general top-down algorithms for fuzzy decision trees induction.

In addition, this technique can be applied without loss of generality to other learning algorithms including rule extraction algorithms. Although this study focused on HEP applications for interpretability purposes, the proposed techniques can be applied to any problem and more especially in the context of other experimental sciences.

We plan to deepen the evaluation of this type of method by comparing the built-in feature construction with the strategy of building one or more features upstream of the induction of the tree. The results may indeed differ from the point of view of performance but also of interpretability.

Acknowledgments

We thank all CLAS12 collaborators for the data simulation and reconstruction software.

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[2] J. R. Quinlan, "Induction of decision trees," *MACH. LEARN*, vol. 1, pp. 81–106, 1986.

[3] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2018, pp. 80–89.

[4] D. Chen, G. Goyal, R. S. Go, S. A. Parikh, and C. G. Ngufor, "Improved interpretability of machine learning model using unsupervised clustering: Predicting time to first treatment in chronic lymphocytic leukemia," *JCO Clinical Cancer Informatics*, no. 3, pp. 1–11, 2019.

[5] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, no. 2, pp. 125–139, Jan. 1995.

[6] C. Marsala and B. Bouchon-Meunier, "Construction Methods of Fuzzy Decision Trees," in *JCIS'98*, Durham, North Carolina, United States, Oct. 1998, pp. 17–20.

[7] D. Dubois, H. M. Prade, and J. C. Bezdek, *Fuzzy Sets in Approximate Reasoning and Information Systems*. Norwell, MA, USA: Kluwer Academic Publishers, 1999.

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

[9] B. Bouchon-Meunier, C. Marsala, and M. Ramdani, "Learning from Imperfect Data," in *Fuzzy set methods in information engineering : a guided tour of applications*. John Wiley and Sons, 1997.

[10] B. Bouchon-Meunier and C. Marsala, "Measures of Discrimination for the Construction of Fuzzy Decision Trees," in *FIP'03 - International Conference on Fuzzy Information Processing*, Beijing, China, Mar. 2003, pp. 709–714.

[11] H. Tanaka, T. Okuda, K. Asai *et al.*, "Fuzzy information and decision in statistical model," *Advances in Fuzzy Sets Theory and Applications*, pp. 303–320, 1979.

[12] C. Z. Janikow, "Fuzzy decision trees: issues and methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 1, pp. 1–14, Feb 1998.

[13] B. Chandra and P. Varghese, "Fuzzifying Gini Index based decision trees," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8549–8559, 2009.

[14] C. Olaru and L. Wehenkel, "A complete fuzzy decision tree technique," *Fuzzy Sets and Systems*, vol. 138, no. 2, pp. 221–254, Sep. 2003.

[15] P. Espejo, S. Ventura, and F. Herrera, "A Survey on the Application of Genetic Programming to Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 2, pp. 121–144, Mar. 2010.

[16] I. Swesi and A. Bakar, *Recent Developments on Evolutionary Computation Techniques to Feature Construction*, 01 2020, pp. 109–122.

[17] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 1, pp. 89–99, Feb. 2005.

[18] M. G. Smith and L. Bull, "Genetic Programming with a Genetic Algorithm for Feature Construction and Selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, Sep. 2005.

[19] P. Sondhi, "Feature Construction Methods: A Survey," p. 8, 2009.

[20] K. Krawiec, "Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery Tasks," p. 15, 2002.

[21] F. E. B. Otero, M. M. S. Silva, A. A. Freitas, and J. C. Nievola, "Genetic Programming for Attribute Construction in Data Mining," in *Proceedings of the 6th European Conference on Genetic Programming*, ser. EuroGP'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 384–393.

[22] K. Neshatian and M. Zhang, "Genetic Programming and Class-Wise Orthogonal Transformation for Dimension Reduction in Classification Problems," in *Genetic Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 242–253.

[23] B. Xue, M. Zhang, Y. Dai, and W. N. Browne, "PSO for feature construction and binary classification." ACM Press, 2013, p. 137.

[24] D. Gavrilis, I. G. Tsoulos, and E. Dermatas, "Selecting and constructing features using grammatical evolution," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1358–1365, Jul. 2008.

[25] S. Markovitch and D. Rosenstein, "Feature generation using general constructor functions," *Machine Learning*, vol. 49, no. 1, pp. 59–98, Oct 2002.

[26] U. Khurana, D. Turaga, H. Samulowitz, and S. Parthasrathy, "Cognito: Automated feature engineering for supervised learning," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, Dec 2016, pp. 1304–1307.

[27] D.-S. Yang, L. Rendell, and G. Blix, "A Scheme for Feature Construction and a Comparison of Empirical Methods," p. 6, 1991.

[28] A. Ittner and M. Schlosser, "Discovery of Relevant New Features by Generating Non-Linear Decision Trees," p. 6, 1996.

[29] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-Task Feature Learning," in *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 41–48.

[30] A. Ekárt and A. Márkus, "Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms," *AI EDAM*, vol. 17, no. 03, Aug. 2003.

[31] F. Maes, P. Geurts, and L. Wehenkel, "Embedding Monte Carlo Search of Features in Tree-Based Ensemble Methods," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7523, pp. 191–206.

[32] D. García, A. González, and R. Pérez, "A feature construction approach for genetic iterative rule learning algorithm," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 101–117, Feb. 2014.

[33] D. Garcia, D. Stavrakoudis, A. Gonzalez, R. Perez, and J. B. Theocharis, "A Fuzzy Rule-Based Feature Construction Approach Applied to Remotely Sensed Imagery," in *Proceedings of the 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology*. Gijón, Spain.: Atlantis Press, 2015.

[34] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Int. Res.*, vol. 2, no. 1, pp. 1–32, Aug. 1994.

[35] C. T. Yildiz and E. Alpaydin, "Omnivariate decision trees," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1539–1546, Nov 2001.

[36] 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)*, 2019, pp. 1651–1659.

[37] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, ser. Complex adaptive systems. Cambridge, Mass: MIT Press, 1992.

[38] M. Cintra, M.-C. Monard, and H. Camargo, "Fuzzydt- a fuzzy decision tree algorithm based on c4.5," in *Proceedings of the 2nd Brazilian Congress on Fuzzy Systems (CBSF)*, 01 2012, pp. 199–211.