Interpretable Machine Learning with Bitonic Generalized Additive Models and Automatic Feature Construction

Noëlie Cherrier, Michael Mayo, Jean-Philippe Poli, Maxime Defurne, Franck Sabatié

Interpretable Machine Learning with Bitonic Generalized Additive Models and Automatic Feature Construction

Noëlie Cherrier[1,2], Michael Mayo[3], 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.
[3] Department of Computer Science, University of Waikato, New Zealand.

April 22, 2024

Abstract

In many machine learning applications, interpretable models are necessary for the sake of trust or for further understanding the patterns in the data. In particular, scientists often want models that elucidate knowledge and therefore may lead to new discoveries. Currently, Generalized Additive Models (GAM) are gaining interest in other application domains because of their ability to fit the data well while at the same time being intelligible. Moreover, prior domain-specific knowledge is often valuable to guide the learning. In this work, extensions and generalizations of GAM are proposed to incorporate prior knowledge during the learning phase. Specifically, the fitting method for GAM is modified so that it can fit the data with bitonic functions. In physics for instance, the most discriminative variables often present specific distributions with respect to the target variable, especially peaking (i.e. bitonic) distributions. An algorithm is also described to build automatically bitonic high-level features to be used in the GAM terms. Experiments on three physics datasets are used to validate these ideas in conjunction with physics scientists.

Keywords: Bitonicity, Generalized Additive Models, Experimental physics.

1 Introduction

A common obstacle in machine learning (ML) comes from a tradeoff between performance and interpretability. Several application domains require interpretability to be able to use ML models, for instance in healthcare [1]. In scientific domains in particular, interpretable models are needed for validation on real data and hopefully for knowledge discovery. Arrieta et al. [2] notably provide a definition for interpretability as "the ability to explain or to provide the meaning in understandable terms to a human". This work focuses on producing models interpretable by an expert.

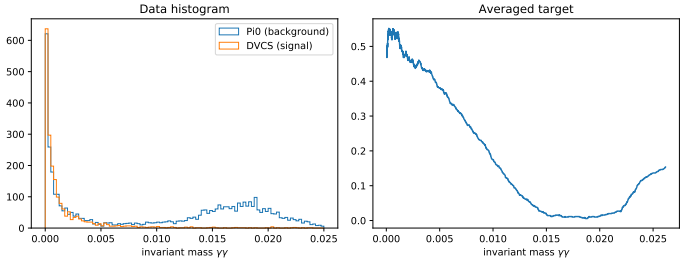


Figure 1: Invariant mass $\gamma\gamma$, a high-level variable often used to recognize π^0 production events. On the left, the unnormalized distributions of the two classes with respect to the invariant mass $\gamma\gamma$. On the right, the averaged target (corresponding to the ratio between numbers of signal and background instances per bin).

Generalized Additive Models [3] (GAM) are often considered as both intelligible and well performing models [4]. The predicted variable in a GAM is a sum of smooth functions of the input variables. The final model is interpreted by observing the inferred smoothed functions of each input variable independently. However, GAM must meet a few requirements to remain interpretable, as they may still be overly complex [2]. According to Arrieta et al., the variables and smooth functions of the GAM must be constrained "within human capabilities for understanding". To fulfill this need, prior knowledge should be incorporated about the problem.

For illustrative purposes, this work focuses on high-energy physics (HEP) problems. Generally, a preprocessing step of feature engineering is performed manually based on domain expertise. In HEP, quantities related to energy, mass or momentum balances which are 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 this way, interpretable FC is performed in this work to determine the discriminative variables of interest to be used in GAM.

In addition, prior knowledge on the expected distributions of the features can be integrated during the inference of GAM terms. A monotonicity assumption is often made in the literature [5, 6, 7, 8]: one or several input variables are assumed to be monotonic with respect to the target variable. The ML model is then constrained to respect this assumption such that the predicted value of the model should be monotonic with respect to the input variable(s). However, we observe that in the HEP field in particular, the most frequently used high-level variables often present a local extremum (see Figure 1 for instance). Other applications can also benefit from bitonicity constraints, such as dose–response analysis [9]. Bitonicity is introduced in this work as an extension to monotonicity and enforced in GAM terms in the context of HEP applications.

The contributions of this work can be summarized as follows: firstly, a definition of bitonicity and an algorithm to verify the bitonicity of a distribution is presented (section 2); secondly, a method to constrain GAM terms to be bitonic is described

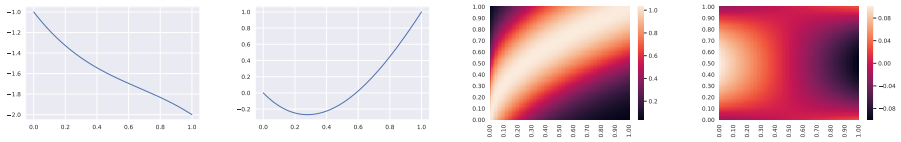


Figure 2: First two plots: two bitonic univariate functions. Third plot: bivariate bitonic function (i.e. bitonic w.r.t its two variables). Fourth plot: non bitonic bivariate function, since the y variable is increasing then decreasing for low x and the opposite for high x.

in section 3, including a bitonic FC algorithm. Several experiments are performed in section 4 to validate the method. The overall interpretability of the generated models is discussed in section 5.

2 Definition and verification of bitonicity

A function f of one real variable is commonly said to be monotonic if and only if $f(y) \geq f(x)$ (resp. $f(y) \leq f(x)$ for the decreasing case) for any (x, y) in $\mathbb{R}$ such that $y \geq x$. Canini et al. [5] define a multi-variable function to be monotonic with respect to feature d if and only if $f(y) \geq f(x)$ (resp. $f(y) \leq f(x)$) for any two feature vectors x, y in $\mathbb{R}^D$ such that $y[d] \geq x[d]$ and $y[m] = x[m]$ for $m \neq d$ with $m, d \in \{1, 2, ..., D\}$.

Definition 1 *f is positively (resp. negatively) bitonic w.r.t. feature d if and only if for each set of values $[x_m]_{m \neq d}$ (setting all values of input X except feature d), it exists at most one x_d^* in the domain of feature d such that these two conditions are satisfied:*

- *$f(X) \geq f(X')$ (resp. $f(X) \leq f(X')$) with $X = (x_1, ..., x_d, ..., x_D)$ and $X' = (x_1, ..., x'_d, ..., x_D)$ for each x_d and x'_d such that $x_d \leq x'_d < x_d^*$,*
- *$f(X) \leq f(X')$ (resp. $f(X) \geq f(X')$) with $X = (x_1, ..., x_d, ..., x_D)$ and $X' = (x_1, ..., x'_d, ..., x_D)$ for each x_d and x'_d such that $x_d^* < x_d \leq x'_d$.*

In contrast to the usual definition of bitonicity in the context of bitonic sorters, the circular shifts are here not taken into account. In addition, this definition includes fully monotonic functions, e.g. if the value of x_d^* is beyond the range of feature d. Quasi-convex and quasi-concave functions are also bitonic functions (the reciprocal is false for $D > 1$). *Unimodality* is a similar term mostly used for distributions. Figure 2 displays two examples of univariate bitonic functions, one example of a bivariate bitonic function and one bivariate non bitonic function.

The bitonicity of a function is often hard to prove with an analytical method, except for simple functions [10]. To numerically quantify the non-bitonicity degree of a function, it is sampled into an ordered vector (sequence noted s), then compared to a close bitonic sequence. Monotonicity can be assessed by comparing the sequence to its cumulative maximum (for a non-decreasing sequence) or to its cumulative minimum (non-increasing). The i^{th}-component of the cumulative maximum is the maximum

value taken by the sequence between components 0 and i. A bitonic sequence is thus equal to its cumulative maximum in its increasing part and to its cumulative minimum in its decreasing part. Other techniques could be considered such as unimodal regression [11], but the previously described method using cumulatives is preferred because it is simpler to implement and since the objective is to rank the features with respect to each other, not to get the closest bitonic approximation. To summarize, the procedure is as follows:

1. Find the point in the sequence leading to the best bitonic approximation using cumulative minimum and maximum on each side of the point;
2. Compute the integral of the absolute difference between the sequence and its bitonic approximation and normalize it by the length and amplitude of the sequence.

The bitonicity penalty of a sequence varies consequently from 0 to 0.5, the worst case being a sequence of alternating zeros and ones. In practice, the maximum is not reached in the experiments since a prior smoothing of the sequences is performed. The bitonicity of a data feature is checked by looking at the variation of the target values along this feature. However, applying directly the procedure detailed above to evaluate bitonicity can be troublesome: first, data are often noisy; second, the target values take a finite number of values in a classification task. To ensure robustness, the data feature is preprocessed as follows (numbers have been determined empirically):

1. Take the target vector $r = (r_1, ..., r_n)$ sorted along the evaluated feature $f = (f_1, ..., f_n)$. Average the values of r where f takes the same values.
2. A moving average box of size $\frac{n}{10}$ is propagated through the r vector.
3. If $n > 1000$, a median filter of size $\frac{n}{100}$ is applied to r.
4. Then if $n > 10000$, a median filter of size $\frac{n}{1000}$ is applied to r.

Finally, the bitonicity is evaluated on the smoothed r sequence. Figure 6 in subsection 3.5 illustrates examples of this procedure.

3 Bitonic functions and features for GAM

This section recalls the background on GAM and details the method to enforce bitonicity before presenting a summary of the overall approach and describing bitonic FC.

3.1 Enforcing bitonicity of shape functions in GAM

Generalized Additive Models (GAM) [12] are of the form: $g(\hat{y}) = c_0 + \sum_{i \in \mathcal{S}} f_i(x_i)$, where g is the link function and $\mathcal{S}$ the set of features. The shape functions f can be modeled in different ways, for instance using splines [13] fitted with backfitting [3] or penalized iteratively reweighted least squares [14]. Other approaches fit tree ensembles with gradient boosting [4, 15].

A method is proposed hereafter to enforce bitonicity for two types of shape functions without loss of generality: splines and functions learnt by a neural network. The idea in both cases is based on the exploitation of the regularization parameter.

Splines as commonly used in GAM are written $f(x) = \sum_k \beta_k b_k(x)$ with b_k basis functions (B-splines) and β_k the parameters to fit. The fitting of f is done by minimizing the penalized sum of squares: $\min_\beta \left\{ \left\| y - B^T\beta \right\|^2 + \lambda \beta^T P \beta \right\}$ with B the vector of $b_k(x)$ and $\lambda\beta^T P\beta$ a penalty term. P can for instance penalize the differences between adjacent β_k, or the second derivative of the shape function. The larger the λ parameter, the smoother the final function. This smoothing parameter is usually optimized (with respect to a performance metric) by generalized cross-validation (GCV) or restricted maximum likelihood (REML).

$\min_W \{\mathcal{L}(y, f_W(X)) + \lambda \mathcal{R}(W)\}$

The following procedure permits to obtain a bitonic shape function:

1. Fit the shape function f with GCV or REML and retrieve smoothing parameter λ_0.
2. Check the bitonicity of f by applying it to a regular test sequence s spanning the range of x and assess bitonicity of $f(s)$ using the procedure detailed in section 2.
3. If $f(s)$ is bitonic, then accept function f as bitonic. Else set $\lambda = \lambda\mu$ with $\mu > 1$, refit f with imposed λ and go back to step 2.

As long as the test sequence s is large enough to account for the complexity of f, the proposed procedure permits to obtain a bitonic function at the cost of multiple refits. Moreover, this procedure will always converge since an infinite λ leads to a linear function. The choice of μ balances between speed and performance: the performance tends to decrease as λ increases and moves off the optimum found by GCV or REML.

The procedure is similar for shape functions learnt by a neural network. The weights of the network must minimize a cost function plus a regularization term expressed as $\lambda\mathcal{R}(W)$, with $\mathcal{R}$ for instance a $\mathcal{L}^1$ or $\mathcal{L}^2$ regularization of the weights. The intuition behind λ is the same as for spline fitting: the larger the λ, the smoother the resulting function. To enforce bitonicity of the learnt function, the same procedure than for splines is applied but setting the first parameter λ_0 as a hyperparameter (small enough).

An experimental validation of this approach is conducted in the next subsection.

3.1.1 Related work on unimodal regression with splines

Some previous works constrain the shape of splines by adding linear constraints to the optimization problem, producing functions that are increasing, decreasing, convex, concave among others [16]. In [9], data are fitted by unimodal B-splines, namely a function with a single local maximum. However, it requires knowing beforehand the location of the maximum to formulate the linear constraint. One must either try all possible locations of the maximum (because of possible noise, the global maximum may not be the proper one for unimodal regression), or perform a more computationally intensive Bayesian approach. Moreover, the authors do not consider other forms

of bitonicity including monotonic functions and decreasing-increasing functions. In contrast, our approach does not require prior knowledge on the type of bitonicity nor on the optimum location. Taking their approach is more time-consuming because of multiple REML computations: two for each monotonicity type, and twice the number of knots for the two other bitonicity types (the knots correspond to the possible locations of the optimum). Our approach computes REML once and then only solves consecutive penalized least squares problems by increasing λ while the function is not bitonic. Moreover, it can be used with any shape function that supports penalization.

3.2 Validation of the principle to enforce bitonicity of shape functions

The previously presented method is tested on a toy dataset of 1000 instances displayed on Figure 3 (data generated with the `make_classification` function of scikit-learn [17]). Experiments are conducted by fitting a bivariate GAM term, either with splines or with a multilayer perceptron (MLP), while trying different values for the regularization parameter λ. The evolution of the F1 score and the bitonicity penalty obtained by the fitted function is plotted on Figure 4. Figure 5 depicts the fitted terms at small, large, and optimal λ (for which the bitonicity penalty is 0).

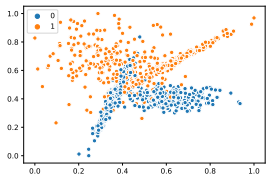


Figure 3: Toy dataset

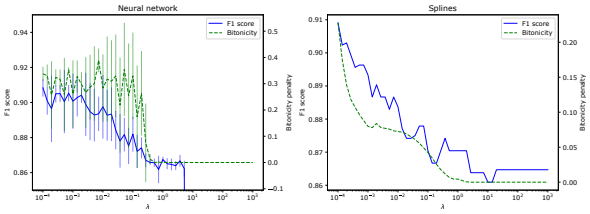


Figure 4: Evolution of F1 score and function bitonicity against regularization parameter λ.

These plots show an experimental validation of the proposed method, as the bitonicity decreases until reaching 0 definitively as λ increases. Moreover, these plots give an interesting comparison between spline and neural network fitting methods: while the spline terms never obtain a score below 0.8, the score of the neural network fitted terms finally drops to 0 (outside the scope of the plot) shortly after reaching bitonicity.

3.3 Building a complete model with gradient boosting

The following algorithm builds a complete GAM with bitonic shape functions. For a binary classification task, a list of inputs X and a target vector y are considered. A standard GAM can be modeled by a sigmoid of the sum $F(x)$ of the GAM terms. In the proposed method, the GAM is built iteratively, building a feature at each step. A first prediction model predicts p_0 for each x in X, p_0 being the proportion of the majority class. The objective is to minimize the cross-entropy between the target y and

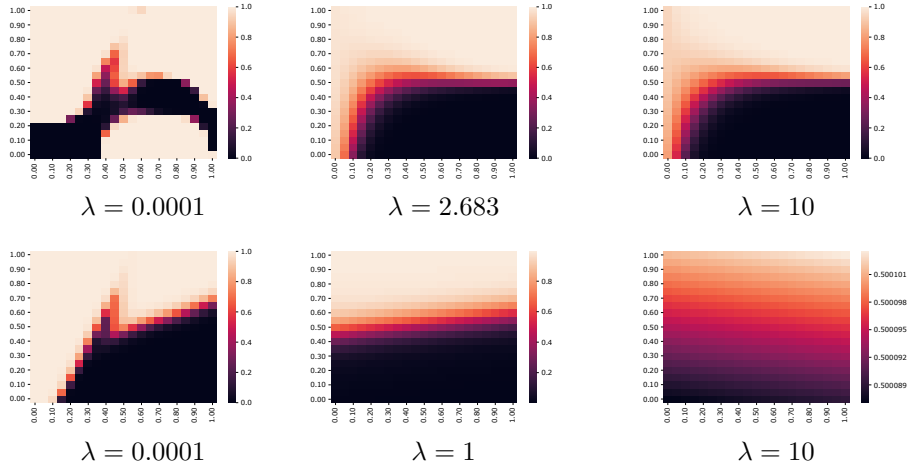


Figure 5: Top: spline models fitted on toy dataset. $\lambda = 2.683$ corresponds to the minimal λ value for which the bitonicity equals 0. Bottom: MLP models fitted on toy dataset. $\lambda = 1$ corresponds to the minimal λ value for which the bitonicity equals 0.

the prediction $p(x)$, noted p:

$$\mathcal{L}(F(x)) = -\left(y \log(p) + (1-y)\log(1-p)\right), \quad p(x) = \frac{1}{1+e^{-F(x)}}. \tag{1}$$

A gradient boosting method is applied to iteratively improve this convex loss function. At the n-th step, a term $h_n(x)$ is added to the current GAM: it is fitted to the current objective while enforcing its bitonicity using the set $\{(x_i, r_{i,n})\}$.

$$r_{i,n} = -\frac{\partial \mathcal{L}_i(F_{n-1}(x))}{\partial F_{n-1}(x)} = y_i - p_{i,n-1}, \quad F_n(x) = F_{n-1}(x) + \alpha_n h_n(x). \tag{2}$$

A gradient descent with its own learning rate β determines the learning rate α_n to minimize the averaged cross-entropy. α_n is directly linked to the importance of the new term in the global result.

The remaining concern is to carefully select the involved variable x in each term h_n. In HEP, relevant variables are often high-level combinations of some base variables, as stated in the introduction. At each step of the boosting algorithm, one feature is built adapted to the ongoing regression problem with the FC algorithm presented in next subsection. The overall process is summarized in Algorithm 1.

3.4 Automatic construction of bitonic features

This subsection proposes an adapted bitonic feature construction (BFC) algorithm. The literature in automatic FC is very abundant and a survey can be found in [18] or [19]. A constrained genetic programming algorithm is proposed in [20] to build interpretable

Algorithm 1: FCGAM algorithm

Input: $data$ used to build the features, of size (m, d), y target vector of length m
n the number of GAM terms to learn

Initialization:

$p \leftarrow p_0$ proportion of the majority class in y, $F \leftarrow \log\left(\frac{p}{1-p}\right)$, $r \leftarrow y - p$

for $i \leftarrow 0$ **to** n `// one iteration builds one term of the GAM`

do

(1) Build one single feature z using FC algorithm with r as the target for the fitness function. Bitonicity may or may not be enforced at this step.

(2) Train a single GAM term on the built feature z with target r. Bitonicity may or may not be enforced at this step.

$h \leftarrow predict(z)$, $g \leftarrow +\infty$, $\alpha \leftarrow$ random(0,100)

while $|g| > \epsilon$ **do**

$\widetilde{F} \leftarrow F + \alpha h$, $\widetilde{p} \leftarrow \frac{1}{1+e^{-\widetilde{F}}}$, $g \leftarrow \frac{1}{D}\sum_k h(\widetilde{p} - y)$, $\alpha \leftarrow \alpha - \beta g$

$F \leftarrow F + \alpha h$, $p \leftarrow \frac{1}{1+e^{-F}}$, $r \leftarrow y - p$

features for HEP applications. One of the contributions is to use a grammar to enforce the respect of physical units during the construction of a new feature.

The constrained FC method of [20] is reused to build features for GAM terms. At a given step n, a single GAM term that fits the target $r_n = y - p_{n-1}$ is added to the model. The fitness function for FC is adapted to better suit the tackled problem of this work. A shallow decision tree with maximum four leafs is trained on the set $\{(z_i, r_{i,n})\}$, z_i being the candidate feature. Thus, the decision tree can only perform cuts on the candidate feature and observe its discriminating power. The fitness of the candidate feature is minus the RMS (Root Mean Squared) error between the prediction of the shallow tree and the target r_n. To enforce bitonicity, a bitonicity penalty term is added to the fitness of z_i. This penalty term b is the result of applying the procedure of section 2 on the sequence of residuals r_n, sorted along z_i. In the end, the fitness of the candidate feature z_i is $f = -(\text{RMS} + b)$, to be maximized during the evolution process.

3.5 Algorithm variants and bitonicity threshold

We consider four variants of Algorithm 1 regarding the bitonicity constraints (summarized in Table 1). Bitonicity can be enforced or not during FC (label `(1)` in Algorithm 1) or for shape functions (label `(2)`).

To activate the bitonicity constraint for shape functions, a parameter b_{max} is set: if the bitonicity of a built feature is below b_{max}, the associated shape function will be forced to be bitonic. Looking at various features from the three datasets used in the experiments, a near-optimal bitonicity threshold b_{max} can be set at 0.04 (see Figure 6).

Table 1: Four variants of Algorithm 1. FCGAM_b_{max} makes the shape functions bitonic if and only if the feature is itself bitonic, i.e. if and only if the bitonicity b of the feature is below b_{max}.

Name	Bitonicity enforced in FC	Bitonicity enforced in shape functions
FCGAM_$\varnothing$	No	No
FCGAM_b_{max}	No	Yes if $b \leq b_{max}$
FCGAM_∞	No	Yes
BFCGAM	Yes	Yes

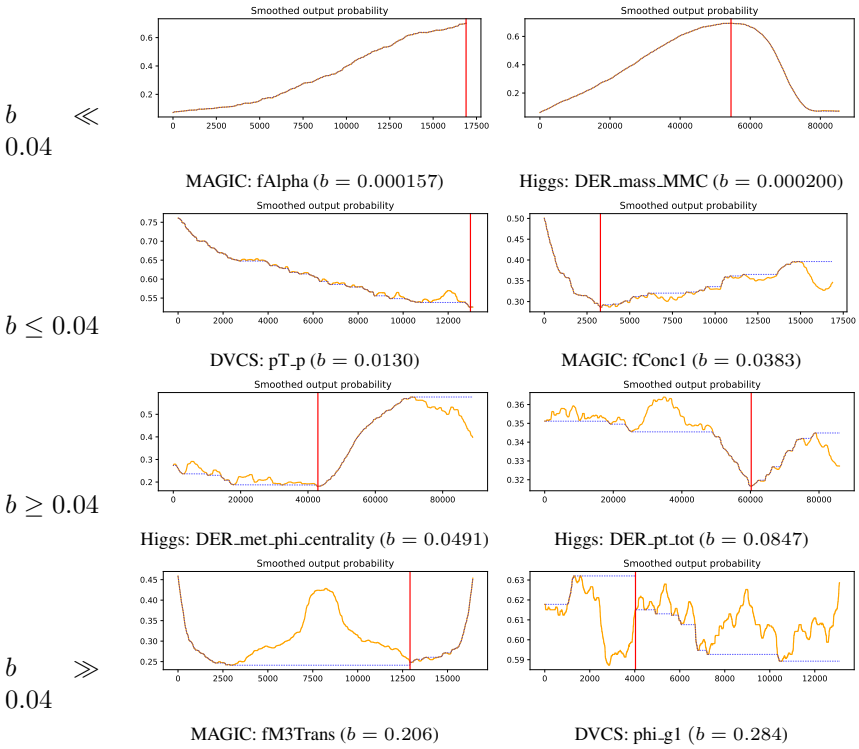


Figure 6: Feature bitonicity examples on all three datasets, from smaller to higher bitonicity penalties. On the graph is plotted the output probability after smoothing (i.e. the vector used for the computation of the bitonicity penalty). In dotted black is the cumulative minimum/maximum that is the reference to compute the difference and get the bitonicity penalty corresponding to the area between the dotted line and the orange feature data. The red vertical line marks the hypothesis of the algorithm for the extremum.

4 Experiments

4.1 Experimental setup

4.1.1 Datasets

Three HEP binary classification problems are considered in this study, with very different experimental setups and studied processes. However the objective is the same: to

isolate signal instances from one or several background sources. Our focus on HEP problems with information about the input variables complicates experiments on a larger number of datasets. The DVCS dataset comprises 30 raw features for 14730 instances, the Higgs dataset [21] 17 raw features for 100000 used instances, the MAGIC dataset [22] 10 raw features for 19020 instances. Detailed descriptions of the DVCS and Higgs datasets are provided in section 5 along with an interpretability discussion.

4.1.2 Shape functions parameters

The GAM version with neural network as shape function uses a MLP regressor with two hidden layers of size 100 each, Adam optimizer with a constant learning rate of 0.001 and rectified linear unit as activation function. The shape (100,100) of the network is not fine-tuned since the objective is to have sufficient degrees of freedom to handle the dimensionality of the problem, while letting the optimization of the regularization parameter (through bitonicity requirement) compensate the potential overfitting. These parameters were found by quick testing on the datasets.

As for the terms modeled with splines, penalized B-splines are used with 14 knots spaced along the quantiles of the feature.

4.1.3 Other hyperparameters

In the FC algorithm, we set the population size to 500 and the number of generations to 70. The multiplying factor μ to increase the regularization λ if the resulting shape function is not bitonic is arbitrarily set to $\mu = \sqrt{10}$. The β parameter for the gradient descent of the learning rates α_n is set to 30 and the demanded maximum ϵ for the gradient is set to 10^{-5}. These parameters have been experimentally proven to lead to convergence in almost all cases for the experimental datasets. In the case an α_n rate was not found, the current iteration n of the FCGAM algorithm is dropped and recomputed (a new FC and shape function fit are done). 10 independent runs for each configuration are performed because of the stochastic nature of genetic programming in FC. The mean and standard deviation of the F1 score are presented for each dataset and algorithm configuration, averaged over the 10 runs and doing 5-fold cross-validation (50 runs in total for each numerical result).

4.2 Performance comparison

Table 2 presents the results obtained while applying the four variants detailed in subsection 3.5 on the three datasets using either splines or MLP as GAM shape functions, along with three baselines. A first observation is that the FCGAM algorithm gives better results than the baselines on DVCS and MAGIC datasets. One could have expected the opposite, since more complex and less interpretable algorithms such as a neural network or XGBoost are supposed to perform better on complex problems, which is the case for the Higgs dataset.

Apart from the baselines comparison, the fourth version (BFCGAM) always gives worse scores than the no bitonicity FCGAM_$\varnothing$ version, in a significant manner for the Higgs dataset in particular. In all cases, letting the FC free and enforcing bitonicity

Table 2: F1 score (mean and standard deviation over 10 runs for each fold in a 5-fold cross-validation) of the proposed methods on three datasets, compared with a few baselines. The best score for each dataset is in bold font, while the best variant for splines and for MLP is underlined.

	DVCS	Higgs	MAGIC
Baselines			
(100,100) MLP	0.751 ± 0.007	$\mathbf{0.673 \pm 0.022}$	0.795 ± 0.012
XGBoost	0.772 ± 0.005	0.587 ± 0.002	0.797 ± 0.009
GAM with boosted DT	0.748 ± 0.004	0.539 ± 0.002	0.781 ± 0.008
FCGAM with splines			
FCGAM_$\varnothing$	0.792 ± 0.012	$\underline{0.626 \pm 0.015}$	$\underline{0.806 \pm 0.011}$
FCGAM_b_{max}	$\underline{\mathbf{0.793 \pm 0.006}}$	0.625 ± 0.013	$\underline{0.806 \pm 0.011}$
FCGAM_∞	0.788 ± 0.007	0.609 ± 0.025	0.804 ± 0.012
BFCGAM	0.789 ± 0.007	0.545 ± 0.031	0.792 ± 0.012
FCGAM with (100,100) MLP			
FCGAM_$\varnothing$	$\underline{0.792 \pm 0.009}$	0.627 ± 0.013	$\underline{\mathbf{0.808 \pm 0.012}}$
FCGAM_b_{max}	$\underline{0.792 \pm 0.008}$	$\underline{0.628 \pm 0.013}$	0.807 ± 0.011
FCGAM_∞	$\underline{0.792 \pm 0.006}$	0.625 ± 0.013	0.807 ± 0.012
BFCGAM	0.788 ± 0.005	0.529 ± 0.057	0.785 ± 0.012

only on relevant shape functions (i.e. those for which the feature is actually bitonic) in the FCGAM_b_{max} variant improves the score at the level of the FCGAM_$\varnothing$ version. Even if not significantly, the FCGAM_b_{max} version with $b_{max} = 0.004$ threshold sometimes gets better results than the FCGAM_$\varnothing$. One conclusion for this is that forcing bitonicity may be good for interpretability (this will be discussed in section 5), but can be too restrictive: some really discriminative features are not bitonic and are indispensable to get a good score. Next subsection discusses the bitonicity potential of the datasets and why some datasets behave well under bitonicity constraint while others do not.

4.3 Bitonicity potential of the different datasets

Enforcing bitonicity on built features or shape functions will only be beneficial if there exist a discriminative set of bitonic features for a given dataset. Figure 7 displays the boxplots of the features bitonicities: those already present in the dataset (raw features) and those which have been found to be discriminative through the FC process (built features) without the bitonicity constraint. Therefore, the built features boxplots represent well the distribution of the most discriminative features for a given dataset. These plots have been made with all the features built in the FCGAM_$\varnothing$ configuration, so around 1000 built features and around 10 raw features for each dataset.

The bitonicity of the raw features seems not to influence the bitonicity of the built high-level features nor the potential of a dataset to get good results while enforcing

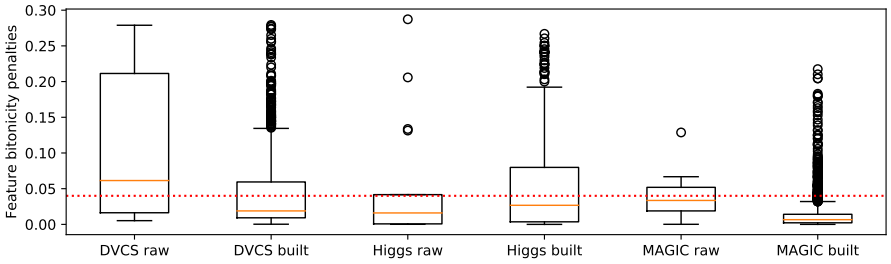


Figure 7: Boxplots of features bitonicities. The box indicates the first and third quantiles with an orange line at the median, while the whiskers extend to the farthest data point within 1.5 IQR (interquartile range) after the box. The horizontal dotted red line represents the bitonicity threshold b_{max} set to 0.04 to trigger the bitonicity constraint on the shape function.

bitonicity. Indeed, the Higgs dataset presents mainly bitonic raw features whereas the scores are impaired under bitonicity constraints, when it is the opposite for DVCS.

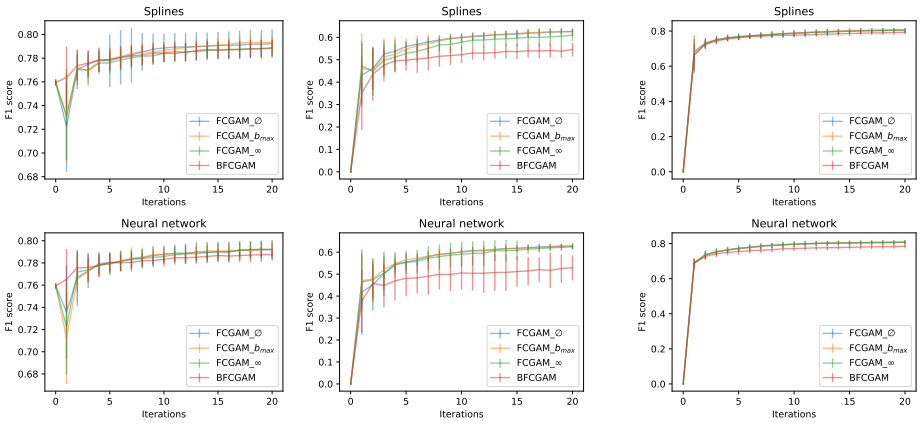
The large majority of the features built for the MAGIC dataset are bitonic, without the need to add a bitonicity penalty term during the FC process. It is then logical that the scores of the MAGIC dataset are not penalized when adding this constraint which is already satisfied. However, the most discriminative features found for the Higgs dataset (i.e. the built features) are often not bitonic, hence the decrease in score when trying to force the bitonicity. Not all the features built for the DVCS dataset are bitonic, however the bitonicity penalty does not impair the performance on this dataset. Some redundancy may indeed be present between the built features, hence all the required information to perform a proper classification can be contained in the subsample of bitonic features.

5 Discussion on interpretability

The goal here is to assess the interpretability of the trained models from the point of view of experts in the field. Notably, an explicit textual explanation of the model is not needed. Indeed, expert physicists are able to interpret high-level built features (physical formulas) and shape functions (can be viewed as function curves).

The global GAM consists of 20 terms, namely 20 shape functions each associated to a built feature. Figure 8 displays the evolution of the classification score for each dataset against the number of GAM terms. Interpretability decreases with increasing number of GAM terms hence increasing number of features. This is a trade-off that experts must consider depending on their own criteria.

The focus now is on the DVCS and Higgs datasets only since we are not experts in the MAGIC classification problem. An analysis of one feature and the associated fitted shape function for each of the DVCS and Higgs datasets is performed hereafter. Figures 9 and 10 are presented in the same way: the left plot is the target vector binned along



DVCS dataset Higgs dataset MAGIC dataset

Figure 8: Convergence curves for all three datasets with splines (on the top) or neural network (on the bottom) as shape function. The evolution of the F1 score is plotted against the number of iterations in the boosting algorithm (Algorithm 1).

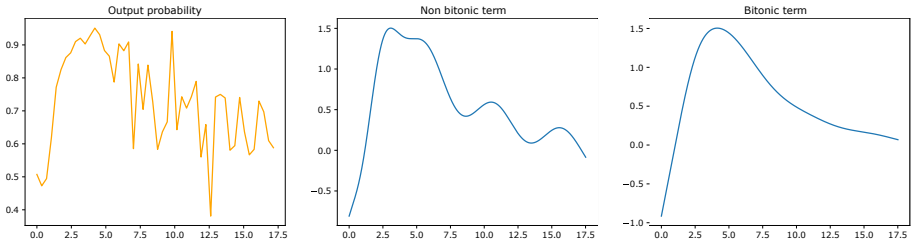


Figure 9: DVCS: $\angle(p_{\gamma_2}, p_{\gamma_1} + p_{\gamma_2})$. Lower value means higher probability to have a signal event.

the built feature (so the y value on the plot is the averaged target for a bin), the central plot is a GAM term learnt for this feature using splines without bitonicity enforcement, the right plot is a GAM term learnt using splines and with bitonicity enforcement. For the DVCS and Higgs datasets, the physical problem is first explained followed by an interpretation of a frequent GAM term.

5.0.1 DVCS dataset

At Jefferson Laboratory, an electron beam scatters off protons. The objective is to discriminate between the γ^{DVCS}-events whose final state is composed of an electron, a proton and a photon noted γ, and 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) and angles are available for each identified particle.

Figure 9 illustrates a DVCS built feature: the angle between γ_2 the lowest energetic

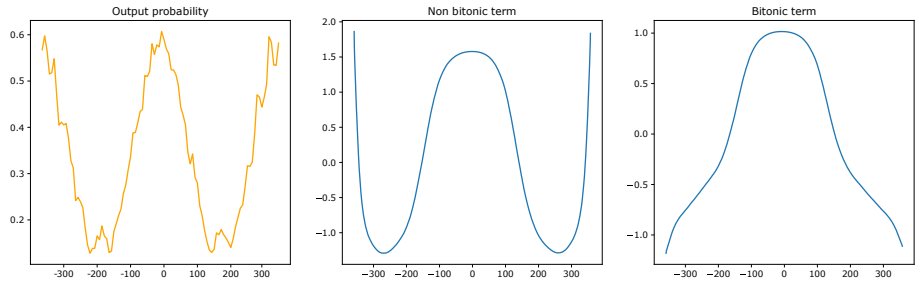


Figure 10: Higgs dataset: $\phi_{lep} - \phi_{missingtE}$. Higher value means higher probability to have a signal event.

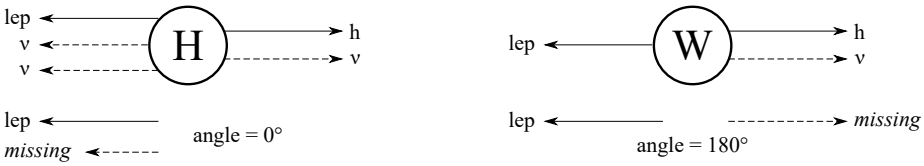


Figure 11: Illustration of signal events on the left and one type of background events on the right.

photon and the sum of two detected photons $\gamma_1 + \gamma_2$. A signal γ^{DVCS}-event involves a single γ photon. But an uncorrelated photon from background may be simultaneously detected. It then 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, the distribution of this angle is not random and presents a peak around 5 degrees. However, the oscillations in the non bitonic term are probably learnt from the noise present in the data. The bitonic term permits to solve this irregularity: experts can visually tell that it generalizes better and is more consistent with their expectations.

5.0.2 Higgs dataset [21]

At CERN, Higgs particles are notably produced out of the collisions of two protons. The objective of the dataset is to detect Higgs bosons decaying into two τ-particles. Geometrical features are available for each detected particle.

The angle between the lepton and a hypothetical missing particle illustrated by Figure 10 is one of the most common feature built by a FCGAM for the Higgs dataset. Indeed this missing momentum actually relates to undetectable particles called neutrinos. In signal events, the neutrinos are in majority emitted in the same direction than the lepton. However, in several background processes, only one neutrino is emitted in the opposite direction of the lepton (see Figure 11). Therefore, the probability to have a signal event is higher at 0 degrees and at its lowest at ± 180 degrees. This feature is highly discriminative but not bitonic. Enforcing bitonicity on this feature is counterproductive.

6 Conclusion

GAM are widely considered as intelligible models, suitable for applications where transparency and expert interpretation is needed. In this work, bitonicity is introduced to take into account prior knowledge about HEP applications. A method is proposed to test the bitonicity of a feature and to enforce it when fitting shape functions. Feature construction is also incorporated in the process since raw variables in HEP are often not the most relevant for classification purposes and since interpretable models often lack of sufficiently complex internal representation of data.

Experiments on three HEP datasets show that enforcing bitonicity on terms associated with bitonic features increases the interpretability potential and generalization power of the global model, with a performance classification score comparable to the score obtained without constraint, if not greater. However, some datasets have shown to be more adapted to the bitonicity approach, depending on the bitonicity degree of the most discriminative features for the classification problem.

In future work, we plan to deepen our studies on the reasons why bitonicity is working better on some datasets than on others. In addition, we started conducting experiments including 2D terms that involve pairwise interactions between features to complete our existing studies on univariate GAM terms.

Acknowledgments

We would like to thank the CLAS12 collaboration for the simulation software.

References

[1] Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L.: Explaining explanations: An overview of interpretability of machine learning. In: 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA). pp. 80–89. IEEE (2018)

[2] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion **58**, 82–115 (2020)

[3] Hastie, T., Tibshirani, R.: Generalized additive models. Statist. Sci. **1**(3), 297–310 (08 1986)

[4] Lou, Y., Caruana, R., Gehrke, J.: Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 150–158 (2012)

[5] Fard, M.M., Canini, K., Cotter, A., Pfeifer, J., Gupta, M.: Fast and flexible monotonic functions with ensembles of lattices. In: Advances in Neural Information Processing Systems. pp. 2919–2927 (2016)

[6] Fard, M.M., Canini, K., Cotter, A., Pfeifer, J., Gupta, M.: Fast and flexible monotonic functions with ensembles of lattices. In: Advances in Neural Information Processing Systems. pp. 2919–2927 (2016)

[7] Gupta, M., Cotter, A., Pfeifer, J., Voevodski, K., Canini, K., Mangylov, A., Moczydlowski, W., Van Esbroeck, A.: Monotonic calibrated interpolated look-up tables. The Journal of Machine Learning Research **17**(1), 3790–3836 (2016)

[8] Nguyen, A.p., Martínez, M.R.: Mononet: Towards interpretable models by learning monotonic features. arXiv preprint arXiv:1909.13611 (2019)

[9] Köllmann, C., Bornkamp, B., Ickstadt, K.: Unimodal regression using bernstein–schoenberg splines and penalties. Biometrics **70**(4), 783–793 (2014)

[10] Barthelemy, T.: On the unimodality of METRIC Approximation subject to normally distributed demands (2015)

[11] Stout, Q.F.: Unimodal regression via prefix isotonic regression. Computational Statistics & Data Analysis **53**(2), 289–297 (2008)

[12] Lou, Y., Caruana, R., Gehrke, J., Hooker, G.: Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 623–631 (2013)

[13] Wood, S.N.: Thin-plate regression splines. Journal of the Royal Statistical Society (B) **65**(1), 95–114 (2003)

[14] Wood, S.: Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC, 2 edn. (2017)

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

[16] Pya, N., Wood, S.N.: Shape constrained additive models. Statistics and computing **25**(3), 543–559 (2015)

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

[18] Sondhi, P.: Feature construction methods: a survey. sifaka. cs. uiuc. edu **69**, 70–71 (2009)

[19] Swesi, I.M.A.O., Bakar, A.A.: Recent developments on evolutionary computation techniques to feature construction. In: Asian Conference on Intelligent Information and Database Systems. pp. 109–122. Springer (2019)

[20] Cherrier, N., Poli, J.P., Defurne, M., Sabatié, F.: Consistent feature construction with constrained genetic programming for experimental physics. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 1650–1658. IEEE (2019)

[21] Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kegl, B., Rousseau, D.: Learning to discover: the Higgs boson machine learning challenge - Documentation (2014)

[22] Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml