Interpretable machine learning for CLAS12 data analysis

Noëlie Cherrier

▶ **To cite this version:**

Noëlie Cherrier. Interpretable machine learning for CLAS12 data analysis. High Energy Physics - Experiment [hep-ex]. Université Paris-Saclay, 2021. English. NNT : 2021UPASP017 . tel-03184811

HAL Id: tel-03184811
https://theses.hal.science/tel-03184811

Submitted on 29 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université
PARIS-SACLAY

Interpretable Machine Learning for CLAS12 Data Analysis

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 576, Particules, Hadrons, Énergie, Noyau, Instrumentation, Imagerie, Cosmos et Simulation (PHENIICS)
Spécialité de doctorat: Physique hadronique
Unité de recherche: Université Paris-Saclay, CEA, Département de Physique Nucléaire, 91191, Gif-sur-Yvette, France
Référent: Faculté des sciences d'Orsay

Thèse présentée et soutenue à Saclay, le 1er mars 2021, par

Noëlie CHERRIER

Composition du jury:

David Rousseau Directeur de recherche, Université Paris-Saclay	Président du jury
Christophe Denis Maître de conférences, HDR, Sorbonne Université	Rapporteur & Examinateur
David Ireland Professeur, University of Glasgow	Rapporteur & Examinateur
Michael Mayo Maître de conférences, University of Waikato	Examinateur
Franck Sabatié Directeur de recherche, Université Paris-Saclay	Directeur de thèse
Maxime Defurne Ingénieur de recherche, Université Paris-Saclay	Co-encadrant
Jean-Philippe Poli Ingénieur de recherche, Université Paris-Saclay	Co-encadrant

Thèse de doctorat

NNT: 2021UPASP017

Remerciements / Acknowledgements

Après la finalisation du manuscrit puis la soutenance deux mois plus tard, me restent deux tâches importantes : rédiger le résumé en français de 5 à 10 pages, et écrire mes remerciements. Je suis heureuse de pouvoir remercier ici les personnes m'ayant aidée et/ou soutenue pendant ces trois ans (et quatre mois) de thèse.

I start by thanking the members of my jury for accepting to be part of it, and for the very interesting questions during the defense and their suggestions for further work. Thank you in particular to the two reviewers, David Ireland and Christophe Denis, for their comments on my manuscript. Thank you to David Rousseau for accepting to be the president of this jury.

Cette thèse n'aurait jamais existé sans la coopération de mes trois formidables encadrants : Franck, Jean-Philippe et Maxime. Pour commencer, merci d'avoir imaginé ce sujet de thèse pour lequel j'ai eu un intérêt immédiat lorsque j'ai trouvé l'annonce sur le site du CEA. Franck, merci d'avoir dirigé cette thèse, laissant toute latitude à Jean-Philippe et Maxime tout en apportant un regard neuf quand nécessaire. Jean-Philippe, Maxime, vous étiez complémentaires dans mon encadrement quotidien. Je me suis beaucoup amusée à confronter vos points de vues et j'ai souvent été surprise des idées originales que vous pouviez proposer concernant votre domaine de non-expertise (!). Merci à vous deux pour votre dévouement et votre disponibilité, quelles qu'étaient vos autres obligations.

I must thank a fourth person who played a significant role in this thesis. In addition to being examiner for my defense, Michael Mayo is the initiator of the research we conducted together on generalized additive models. Thanks to a casual discussion during a conference in Wellington, these models took a big importance in my thesis, as they turned out to be highly appreciated by physicists while obtaining very good results for the physics analysis. Therefore, thank you Michael for introducing me to GAM and for the long discussions and collaboration that followed.

Thanks to the 31 respondents to the survey of chapter 8, the results were numerous and so much interesting to analyze. Many thanks to them for their time, and also special thanks to the first six testers who helped us designing and improving the survey: Francesco Bossu, Nicole d'Hose, David Lhuillier, Loïc Thuilliez, Marine Vandebrouck and Michael Winn. Merci Nicole en particulier pour avoir fait le relais auprès de COMPASS.

Merci à mes deux laboratoires d'accueil : le DPhN et le SID. J'ai pris beaucoup de plaisir à nos discussions notamment lors des pauses du midi, que ce soit parmi des physiciens ou des informaticiens/mathématiciens. Merci à l'ensemble du SID pour leur accueil, leurs discussions et leurs encouragements sur la fin : merci à Eiji, Marine, Youen, Arnaud, Ismaïl, Alyssa, Marisnel, Hung, Baptiste, Rafaël, Sandra... Merci Mikael d'avoir implémenté la construction de features dans ExpressIF®, et Sébastien d'avoir essayé d'extraire les probabilités de transition de livres de physique. Merci Edwin pour avoir recodé FURIA et pour notre travail sur les imprécisions, Camille pour ton aide avec R et les GAM. Côté physiciens, merci évidemment au groupe du midi : Hervé & Hervé, Cédric, Francesco, Valerio, Arek, Aurore, Nabil, Christopher (merci pour tes distractions du matin !). Merci à Danielle et Isabelle sans qui le département aurait bien du mal à tourner. Merci aussi aux fameux précaires du DPhN, et en particulier à Nancy, Zoé et Aude pour continuer à faire vivre ce groupe.

Merci aussi à Benjamin, Charles, Antoine, Loïc. Merci à Guillaume et Marouen pour notre travail commun sur les données CLAS12. Désolée de vous avoir battus ! ☺

Merci à l'ensemble de mes proches pour leur soutien et pour m'avoir offert des moments de décompression bienvenus. Merci à la double paire de jumeaux : Constant, Gautier, Raphaël, Alexandre, et à Marine, Inès et Laure. Merci au Peignch étendu pour nos soirées BGA, merci pour vos encouragements, félicitations à ceux qui ont soutenu et bon courage aux suivants !

Merci à ma famille pour m'avoir soutenu durant ma thèse, et pour avoir sincèrement essayé de comprendre mon sujet (avec plus ou moins de succès pour certains) ! Merci à mes parents et à Rozenn d'être venus en personne à ma soutenance. Merci Gildas d'avoir révisé pour tout comprendre à ma présentation ! Merci Caro pour nos longues discussions par téléphone qui font toujours très plaisir, j'espère que ça continuera ! Merci Gaëlle, Gilles, Alexandre, Lyson pour votre soutien à distance.

Enfin, je ne pourrais pas conclure sans remercier ma moitié, Brian, qui m'a aidé et soutenu sur tous les plans : professionnellement (pour les questions de physique, les scripts, les RDataFrame...), logistiquement (tu fais trop bien la cuisine pour que j'ose interférer) et personnellement (merci d'avoir supporté mes sautes d'humeur et m'avoir aidée à tenir jusqu'au bout). J'espère être à la hauteur pour te rendre la pareille.

Contents

Remerciements / Acknowledgements iii

List of Abbreviations ix

Introduction 1

I Context and positioning 3

1 Study of the proton structure at CLAS12 5
1.1 Introduction to particle physics . . . 6
1.2 Generalized parton distributions . . . 8
1.3 CLAS12 experiment . . . 15
1.4 Methodology for DVCS analysis . . . 20

2 Interpretable machine learning 27
2.1 Interpretability: a general introduction . . . 28
2.2 Common intrinsically interpretable machine learning models . . . 38
2.3 Conclusion . . . 54

3 CLAS12 simulation and baselines using transparent models 55
3.1 Monte Carlo simulation of DVCS and π^0 production events . . . 55
3.2 Baselines using transparent models . . . 60
3.3 Proposed approach . . . 64

II Interpretable machine learning through feature construction 67

4 State of the art of feature construction 71
4.1 Tree-based feature construction algorithms . . . 73
4.2 Evolutionary-based feature construction algorithms . . . 74
4.3 Embedded feature construction . . . 81

5 Interpretable feature construction as a prior method 83
5.1 Constrained feature construction algorithm . . . 84
5.2 Experiments on prior feature construction . . . 88
5.3 Conclusion and perspectives . . . 100

6 Interpretable embedded feature construction 101
6.1 Embedded feature construction in tree-based and sequential covering algorithms . . . 102
6.2 Boosting feature construction in generalized additive models . . . 111
6.3 Conclusion and perspectives . . . 127

III From simulation to real CLAS12 data analysis 131

7 Model transfer to real data 135
7.1 State of the art of domain adaptation . . . 137
7.2 Domain adaptation of the particles momenta . . . 142
7.3 Experiments . . . 146
7.4 Conclusion and perspectives . . . 159

8 Interpretability evaluation by experimental physicists 163
8.1 Survey form . . . 164
8.2 Results and discussion . . . 168
8.3 Conclusion . . . 184

9 Analysis of DVCS data from CLAS12 185
9.1 DVCS event selection in CLAS12 data . . . 186
9.2 π^0 subtraction and asymmetry computation . . . 190
9.3 Optimal selection threshold and asymmetry computation . . . 194
9.4 Comparisons with other techniques . . . 197
9.5 Conclusion and perspectives . . . 201

Conclusion 203

Publications and talks 207

Bibliography 209

A Introduction to fuzzy logic 235
A.1 Principle and operators . . . 235
A.2 Reasoning with fuzzy logic . . . 236
A.3 Fuzzy expert systems . . . 236
A.4 Handling imprecisions . . . 238
A.5 Advantages of fuzzy logic in machine learning . . . 238

B Exploiting data imprecisions 239
B.1 Background on the use of imprecisions in transparent models . . . 240
B.2 Adaptation of crisp and fuzzy C4.5 algorithms to imprecise data . . . 242
B.3 Adaptation of FURIA to imprecise data . . . 244
B.4 Computation of CLAS12 imprecisions . . . 245
B.5 Experiments . . . 247
B.6 Discussion . . . 250

C Experimental datasets 251
C.1 CLAS12 . . . 251
C.2 Higgs . . . 253
C.3 $\tau \rightarrow 3\mu$ . . . 256
C.4 MAGIC . . . 257
C.5 Summary . . . 257

D Model hyperparameters 259

E Additional experiments on embedded feature construction in tree-based models 261

E.1 Fuzzy C4.5: std version . . . 261
E.2 Fuzzy C4.5: Fibo version . . . 263
E.3 CART . . . 263
E.4 AdaBoost . . . 265
E.5 GradientBoosting . . . 265

F Additional experiments on domain adaptation 267
F.1 Experiments with smeared simulated data with flat distributions . . . 267
F.2 Experiments with smeared simulated data with cross-sections . . . 268

G Complete interpretability survey and responses 273

H Asymmetries using additional models 305

I Résumé en français 321

List of Abbreviations

AdaBoost	**Ada**ptive **Boost**ing
CART	**C**lassification **A**nd **R**egression **T**ree
CFF	**C**ompton **F**orm **F**actors
CLAS12	**C**EBAF **L**arge **A**cceptance **S**pectrometer at **12** GeV
DIS	**D**eep **I**nelastic **S**cattering
DVCS	**D**eeply **V**irtual **C**ompton **S**cattering
FURIA	**F**uzzy **U**nordered **R**ule **I**nduction **A**lgorithm
GA	**G**enetic **A**lgorithm
GAM	**G**eneralized **A**dditive **M**odel
GBM	**G**radient **B**oosting **M**achine
GE	**G**rammatical **E**volution
GP	**G**enetic **P**rogramming
GPD	**G**eneralized **P**arton **D**istributions
HEP	**H**igh **E**nergy **P**hysics
ID3	**I**terative **D**ichotomiser **3**
JLab	**J**efferson **Lab**oratory
MLP	**M**ulti**L**ayer **P**erceptron
PSO	**P**article **S**warm **O**ptimization
QCD	**Q**uantum **C**hromo**D**ynamics
QED	**Q**uantum **E**lectro**D**ynamics
RMS	**R**oot **M**ean **S**quare
ROC	**R**eceiver **O**perating **C**haracteristic

Introduction

This thesis aims at analyzing complex experimental physics data using and adapting interpretable machine learning models. In particular, the case study of this thesis is the investigation of the proton structure. The proton, which is a constituent of the nuclei of atoms, is itself made of elementary particles called quarks and gluons. While the nature and interactions of these elementary particles are described by the Standard Model, their combination and evolution inside a proton cannot be derived from the equations. Therefore, experimental programs study the proton inner structure. At CLAS12 in the Jefferson Laboratory, electrons are accelerated up to an energy of 10.6 GeV and sent onto a proton target. In the same way as an electronic microscope, electrons permit here to probe matter. The collision between electrons and protons create several output particles. Around the collision site is the CLAS12 spectrometer, designed to detect the majority of these created or scattered particles and measure their characteristics. From these measurements, the task of the experimental physicists is to identify the nature of the interaction that occurred. Indeed, the objective is to measure the frequency of certain interactions of interest, these measurements being related to the descriptors of the proton structure.

Nowadays, machine learning has become a privileged tool to analyze data. In particular, classification models could be used to identify the different interactions at CLAS12 from the detector responses. However, since the analysis pipeline must be carefully controlled for robustness and peer-validated before the results are published, using so called "black-box" machine learning algorithms for event selection is tedious. Explainable artificial intelligence or interpretable machine learning are rising trends in the past decade, making "transparent" machine learning models increasingly more common. In this thesis, we propose and adapt several transparent machine learning models for the analysis of CLAS12 data. Since this category of models often obtains lower classification performances than models that are more opaque, we develop an automated feature construction algorithm aiming at increasing the discriminative power of the inner data representation of such models. Ideally, we could hope to discover new knowledge from the study of the induced models. Part of the work in machine learning in this thesis is also dedicated to the proper functioning of these models on CLAS12 data.

This thesis is the result of the collaboration between two institutes: CEA Irfu for the experimental physics aspect, and CEA LIST for the artificial intelligence dimension. The objectives are both to improve the physics analysis techniques with machine learning and to make contributions to the field of interpretable machine learning. In addition, the analysis conducted in this thesis is part of a larger study on DVCS data at CLAS12: indeed, the interpretable machine learning techniques proposed in this thesis are destined to be compared to two other approaches. The first approach consists in the standard physics analysis, conducted by Guillaume Christiaens from the University of Glasgow as part of his PhD thesis, while the second approach is

based on neural networks and has been conducted by Marouen Baalouch from CEA LIST during a one-year postdoctoral contract.

Overall, this thesis is divided into three parts. Part I introduces the context of the thesis:

- Chapter 1 justifies the analysis from a physics point of view. It introduces the theoretical physics objective and the experimental setup at CLAS12.
- Chapter 2 presents the field of interpretable machine learning. It reviews the state of the art of this area and details the transparent machine learning models that will be used afterwards.
- Chapter 3 is more practical: it details the data simulation process and gives first baselines using transparent machine learning models.

Part II is dedicated to the improvement of transparent machine learning models through automated feature construction, while adapting the latter to the physics analysis and maintaining interpretability:

- Chapter 4 reviews the state of the art in feature construction.
- Chapter 5 presents the proposed feature construction technique and first experiments performing feature construction ahead of model induction.
- Chapter 6 proposes an extension of the previously proposed feature construction technique by embedding it into the induction of transparent machine learning models. In particular, an enhancement of generalized additive models is proposed to improve their interpretability.

Part III focuses on the specific CLAS12 data analysis. The classification performances of the proposed models must be confirmed on real data and their interpretability validated by domain experts before performing the final analysis:

- Chapter 7 investigates domain adaptation to make up for the discrepancies between simulated data and real data.
- Chapter 8 discusses the perceived interpretability of the proposed models by expert experimental physicists through a survey.
- Chapter 9 conducts the final analysis on real data. Physics observables are extracted from data using the proposed machine learning models, and the results are notably compared with standard analysis techniques.

Part I

Context and positioning

Chapter 1

Study of the proton structure at CLAS12

1.1 Introduction to particle physics 6
1.1.1 From the atom to the Standard Model 6
1.1.2 Quarks and the strong interaction 7
1.2 Generalized parton distributions 8
1.2.1 Introduction to GPD 10
1.2.2 Properties of GPD 10
1.2.3 Accessing GPD through deeply virtual Compton scattering 11
1.2.4 Accessing GPD through π^0 electroproduction 13
1.2.5 Experimental status 14
1.3 CLAS12 experiment 15
1.3.1 Experimental setup 16
1.3.2 Accessible phase space and DVCS event detection 19
1.4 Methodology for DVCS analysis 20
1.4.1 Standard physics analysis: event selection, π^0 subtraction and asymmetry computation 22
1.4.2 Improving event selection with interpretable machine learning 24

While the theoretical physics introduction is a logical prerequisite to an experimental physics analysis, some persons will likely read this thesis without the necessary prerequisites. Hence, this chapter has been written to be as understandable as possible for a layperson. If the reader is looking for more details, the reading of the theses of my predecessors is strongly recommended: Gabriel Charles [2013], Maxime Defurne [2015], Nabil Chouika [2018] and Antoine Vidon [2019]. In addition, the review from Guidal et al. [2013] is advised for complementary information on phenomenology (corresponding to section 1.2).

First, section 1.1 dives into the world of particle physics, presenting the Standard Model and particularly the strong interaction, which is a major actor of the proton structure. Section 1.2 deepens the investigation of the proton structure and introduces the generalized parton distributions and the interaction of interest: deeply virtual Compton scattering (DVCS). Then, section 1.3 presents the CLAS12 experiment aiming notably at studying DVCS. Finally, section 1.4 concludes this chapter by describing the challenges associated with the physics analysis of events collected with the CLAS12 spectrometer.

1.1 Introduction to particle physics

1.1.1 From the atom to the Standard Model

From Antiquity to the present day, numerous scientists investigated the structure of matter in greater and greater detail. With increasing knowledge, the size of the probed constituents decreased. The equipment needed to carry out experiments to discover hidden structures became even more complex.

For instance, Rutherford discovered that matter is essentially made of void and that a tiny positively charged nucleus (its radius being 10000 times smaller than the atom) contains the vast majority of the mass of the atom [Gegier and Marsden, 1909]. His experiment consisted in sending helium nuclei (also called alpha particles) onto a thin gold foil [Gegier and Marsden, 1909]. The nuclei either went through the foil unaffected (proving the majority of void) or were scattered in different directions: from slightly to strongly deviated (when they collide with the nucleus).

Further theoretical models and experiments finally led to the model of an atom composed of a nucleus, formed of *protons* and *neutrons*, and a number of bound electrons (see Figure 1.1). The size of the electron cloud determines the size of the atom. Protons and neutrons, also called *nucleons*, are themselves composed of quarks, which are elementary particles such as the electron. Investigations of the proton structure will be further described in section 1.2.

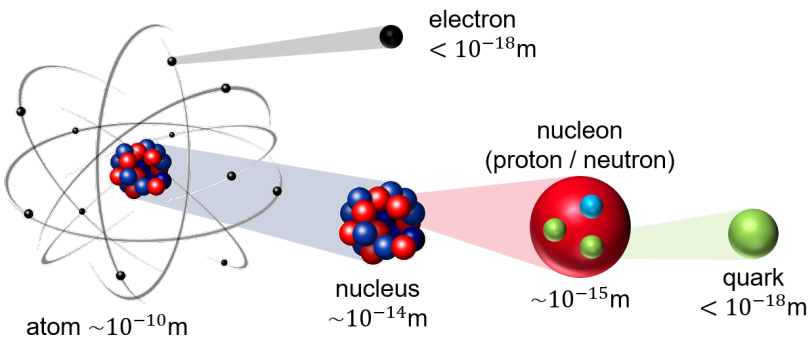


FIGURE 1.1: Inside of an atom. The size of the nucleus relatively to the size of the atom is similar to a big orange compared to the inner Paris.

The rise of particle accelerators since the second half of the 20$^{\text{th}}$ century permitted to discover a wide variety of particles: elementary particles as well as composite particles. Such experiments crosschecked with theoretical advances finally led to the Standard Model: a theory classifying all known elementary particles and describing their interactions (see Figure 1.2). Elementary particles divide into two kinds:

- Quarks (purple on Figure 1.2) and leptons (green on Figure 1.2) that are particles constituting matter. Both categories are subdivided into three families of increasing mass. The particles that make up stable matter are those of the first family (first column on Figure 1.2).
- Gluons, photon, Z and W bosons, i.e. particles that are the vectors of interactions (red on Figure 1.2).

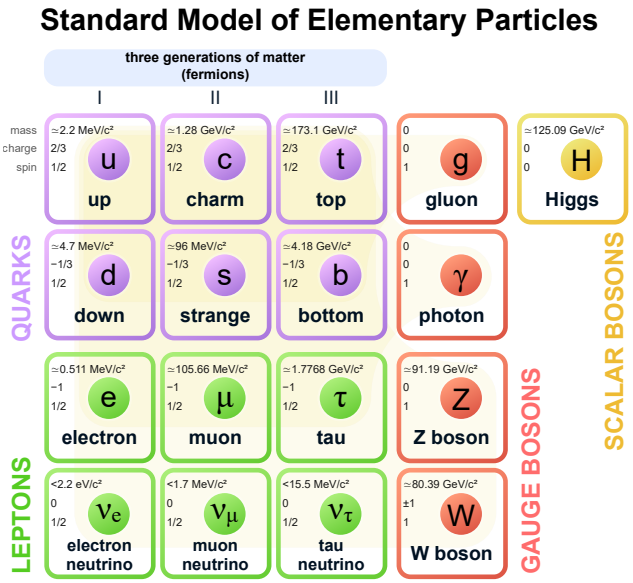


FIGURE 1.2: The Standard Model of elementary particles [Wikipedia, 2020].

The three fundamental interactions of the Standard Model are listed in Table 1.1. The two fundamental interactions that will be encountered in this thesis are the electromagnetic and the strong interaction.

TABLE 1.1: The three fundamental interactions of the Standard Model.

Interaction name	Mediator(s)	Affected elementary particles
Strong interaction	Gluons	Color-charged particles[1] (quarks and gluons only)
Electromagnetic interaction	Photon	Electrically charged particles
Weak interaction	Z, W^+, W^-	Quarks and leptons

1.1.2 Quarks and the strong interaction

Quarks and gluons are the elementary particles affected by the strong interaction. They form a number of composite particles called *hadrons* [Gell-Mann, 1964, Zweig, 1964]. Hadrons are defined by their *valence* quarks, i.e. the quarks responsible for its quantum numbers. For instance, the valence quarks of the proton are two *up* quarks and one *down* quark (uud), and those of the neutron are udd. Those of the meson π^0 are $u\overline{u}/d\overline{d}$ and of the meson π^+ are $u\overline{d}$ ($\overline{u}$ is the anti-particle of the up quark).

The strong interaction is the only fundamental interaction that also affects its mediator particles: the gluons, since they carry a color charge as well. This property makes the strong interaction bind the quarks and gluons together. Its behavior is therefore very different with respect to the electromagnetic and weak interactions. Indeed, to

[1] Color charge for the strong interaction plays here the same role as electric charge for the electromagnetic interaction, except that it decomposes into three dimensions: red, green and blue.

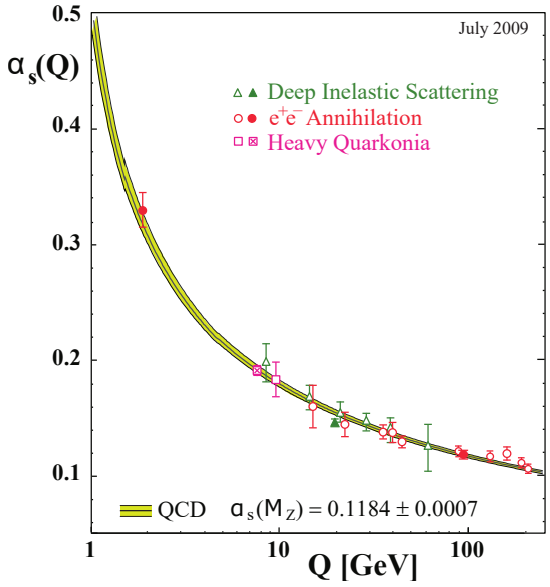


FIGURE 1.3: Measurements of the QCD coupling α_s as a function of energy [Bethke, 2009].

each interaction is associated a coupling constant related to its strength. For instance, the coupling constant of the electromagnetic interaction, described by quantum electrodynamics (QED), is $\alpha_{\text{em}} \approx \frac{1}{137}$. However, the coupling α_s of the strong interaction varies significantly with distance. At long distances (bigger than the nucleon size), it is several orders of magnitude greater than α_{em}. It reduces with decreasing distances (i.e. increasing energy), as plotted on Figure 1.3. Consequently, the large value of α_s at long distances forbids the existence of free quarks. Therefore, quarks and gluons are never detected individually and remain confined inside hadrons.

Most importantly, the equations of QCD are hardly computable when α_s is of the order of 1. Indeed, perturbation theory used to approximate these equations implies to sum terms weighted by a power of α_s. When α_s is small (short distances), quarks are free and perturbation theory applies: high order terms can be neglected. However, when α_s is too large, the equations of QCD are not easily computable anymore.

To compensate for this limit, the field of *hadronic physics* studies the strong interaction in its non-perturbative regime, i.e. where perturbative QCD does not apply. Section 1.2 focuses on the search for *generalized parton distributions* notably through *deeply virtual Compton scattering* (DVCS).

1.2 Generalized parton distributions

Similarly to Rutherford discovering the positive nucleus of the atom, contemporary physicists access proton structure properties through experiments. Following the principle of an electronic microscope, highly energetic electrons are sent to probe a proton: the higher the probe momentum[2], the smaller the probed distance. Cross-sections (i.e. the probability that a given interaction will take place between the electron and the

[2]For particles that have a non-zero mass, the momentum p is proportional to the product of the mass with the velocity of the particle. Globally, the momentum p, mass m and energy E of a particle are linked by the following formula: $E^2 = m^2c^4 + p^2c^2$, with c the speed of light.

proton) can be measured, and properties of the proton can be derived from the underlying theory thanks to these measurements.

A complete description of the structure of the proton would imply the knowledge of both spatial and momentum distributions of the quarks and gluons as well as their correlations. However, the Heisenberg principle prevents the simultaneous knowledge of all these variables. Instead, several processes permit to access different structure functions carrying complementary partial information compatible with the Heisenberg principle. Here, we focus on generalized parton distributions (GPD).

GPD are related to the correlations between the transverse positions and the longitudinal momenta of the partons in the proton (i.e. quarks and gluons). They are accessed through the experimental study of a family of processes called exclusive deep inelastic processes, whose generic Feynman diagram is displayed on Figure 1.4.

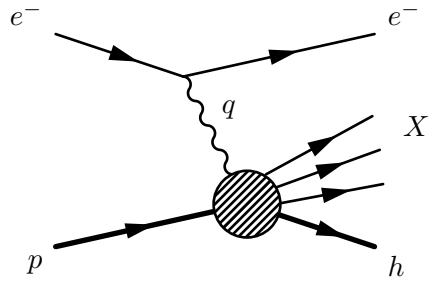


FIGURE 1.4: Feynman diagram of deep exclusive processes. q represents the momentum of the virtual photon. h is the recoil hadron (proton or neutron most of the times) and X the set of particles produced out of the collision. The striped circle represents the soft part that cannot be computed perturbatively (short distance QCD).

Several kinematic variables can be identified using the Feynman diagram of Figure 1.4 and permit to define exclusive deep inelastic processes:

- $Q^2 = -q^2$ is the virtuality of the exchanged photon and must be high for the interaction to be "deep" ($Q^2 \gg M_p^2$, with M_p the mass of the proton). This makes the probed distance much smaller than the size of the proton, therefore the virtual photon interacts with a single constituent (parton) of the proton.
- $W^2 = (p_p + q)^2$ is the available mass in the center of mass frame, with p_p the momentum of the incoming proton and q the momentum of the virtual photon. It must be high as well ($W^2 \gg M_p^2$) so that the interaction is "inelastic", namely several particles are produced out of the interaction (the so-called X set).

In addition, the set of particles X must be defined and measured so that the studied process is exclusive.

Two notable studied processes are DVCS (see 1.2.3) and the deeply virtual meson production (DVMP) (see 1.2.4 for π^0 production, but other mesons can be produced). An introduction to GPD and their properties is proposed before specific insights into DVCS and π^0 production.

1.2.1 Introduction to GPD

There exist eight GPD for each quark flavor and also eight GPD for gluons. These eight GPD (H, E, $\tilde{H}$, $\tilde{E}$, H_T, E_T, $\tilde{H}_T$ and $\tilde{E}_T$) are associated to different helicity[3] combinations of the incoming and outgoing proton and parton.

The GPD are function of four variables: Q^2 the virtuality of the virtual photon (in practice omitted in the notations), x the average longitudinal momentum fraction, t the square momentum transfer to the proton and ξ the longitudinal momentum transfer (see for example Figure 1.6):

$$\xi \simeq \frac{x_B}{2 - x_B}. \tag{1.1}$$

x_B is the Bjorken variable comprised between 0 and 1. In a fixed target experiment (which is the case of CLAS12), it writes:

$$x_B = \frac{Q^2}{2M_p(E - E')} \tag{1.2}$$

with M_p the mass of the proton, E and E' the energies of respectively the incoming and scattered electrons.

1.2.2 Properties of GPD

Extensive information about the proton structure can be derived from the knowledge of GPD. Ralston and Pire [2002] showed that it is possible to plot spatial images of the proton as a function of the longitudinal momentum fraction x thanks to GPD (see Figure 1.5). According to their model, the quarks carrying the largest fraction of momentum would be located close to the nucleon center, while the other quarks and gluons are more spread out.

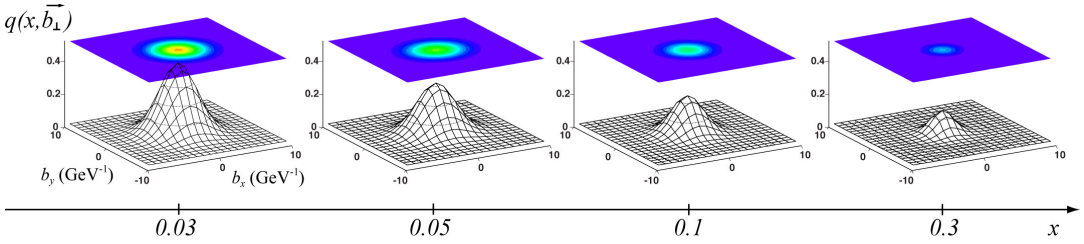


FIGURE 1.5: Simulation of a representation of the nucleon: the transverse position distribution as function of the longitudinal momentum fraction. Figure taken from [Charles, 2013].

Measurements of GPD also give access to the quark contribution to the proton orbital angular momentum [Ji, 1997]:

$$J_i = \frac{1}{2}\int_{-1}^{1} x\left(H_i(x,\xi,0) + E_i(x,\xi,0)\right)dx \quad \forall \xi \in [-1,1] \tag{1.3}$$

with i the quark flavor (or else gluon). J_i is the quark total angular momentum (spin and orbital contributions).

[3] Helicity is the projection of the spin on the direction of the particle's momentum.

1.2.3 Accessing GPD through deeply virtual Compton scattering

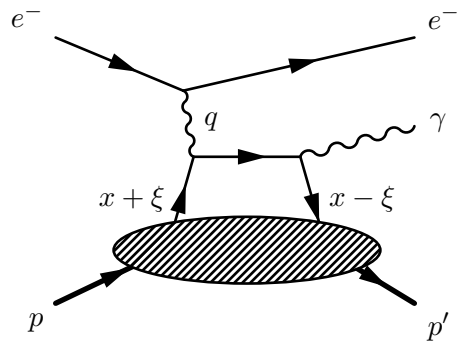


FIGURE 1.6: Feynman diagram of deeply virtual Compton scattering. q is the four-momentum of the virtual photon, x the average longitudinal momentum fraction carried by the quark, ξ the longitudinal momentum transfer. Factorization is illustrated by the dashed horizontal line, above which the perturbative theory applies but below which it is non-perturbative and described by the GPD.

DVCS (Feynman diagram displayed on Figure 1.6) involves the emission of a real photon by a parton (quark or gluon) of the proton from the interaction of the parton with the virtual photon (emitted by the electron).

1.2.3.1 Compton form factors

The variable x of the GPD only exists in the inner cycle of the parton leaving the proton, interacting with the virtual photon and going back into the proton. Contrary to Q^2, ξ and t, x is not observable and must be integrated over. GPD can therefore not be accessed directly.

The DVCS cross-section involves *Compton form factors* (CFF), which are complex convolutions of GPD with the hard scattering process. There is thus one CFF associated to each GPD. For instance:

$$\begin{aligned}\mathcal{H}(\xi,t) &= \int_{-1}^{1} H(x,\xi,t)\, C(x,\xi) dx \\ &= \int_{-1}^{1} H(x,\xi,t) \left(\frac{1}{\xi - x - i\epsilon} - \frac{1}{\xi + x - i\epsilon} \right) dx \qquad \text{at leading order}^4\end{aligned} \tag{1.4}$$

is the CFF associated to the GPD H. Equation (1.4) has been derived following the principle of *factorization*: the hypothesis is made that the interaction can be separated in two parts:

- a hard part (above the dashed line on Figure 1.6) accounting for the scattering between the virtual photon and the parton of the proton; this part can be derived using perturbation theory (function C in equation (1.4));
- a soft part (below the dashed line on Figure 1.6) describing the spectator partons confined in the proton that are not part of the interaction with the virtual photon; this part belongs to the non-perturbative regime described by the GPD.

[4] Higher orders involve perturbations in the Feynman diagram in the form of additional vertices. They appear as additional terms weighted by a coupling constant.

The separation between the real and imaginary parts of the CFF at leading order gives additional insight about the associated GPD. For instance with $\mathcal{H}$:

$$Re\,\mathcal{H}(\xi,t) = \mathcal{P}\int_{-1}^{1}\left(H(x,\xi,t) - H(-x,\xi,t)\right)\left(\frac{1}{x-\xi}+\frac{1}{x+\xi}\right)dx, \tag{1.5}$$

$$Im\,\mathcal{H}(\xi,t) = \pi\left(H\left(\xi,\xi,t\right) - H\left(-\xi,\xi,t\right)\right). \tag{1.6}$$

$\mathcal{P}$ is the principal value of the integral. Therefore, retrieving the imaginary part of the CFF permits to directly access values of the associated GPD at $x = \pm\xi$.

The DVCS amplitude[5] is a linear combination of CFF. They can be separated through a harmonic expansion of the square amplitude as a function of ϕ the angle between the two interaction planes (see Figure 1.7).

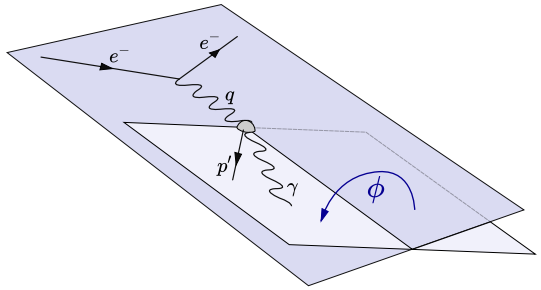


FIGURE 1.7: Geometrical definition of ϕ, the angle between the leptonic plane (interaction $e \to e\gamma^*$) and the hadronic plane (interaction $p\gamma^* \to p\gamma$).

1.2.3.2 Bethe-Heitler interference and DVCS cross-section

The measured cross-section is not directly the square DVCS amplitude, forbidding a direct access to the CFF moduli. Indeed, the *Bethe-Heitler* process (displayed on Figure 1.8) also contributes to photon electroproduction. It has the exact same final state as DVCS: a scattered electron, a recoil proton and an emitted photon. It is actually an elastic scattering between the electron and the proton with a high-energy photon radiated by either the incoming or scattered electron. The Bethe-Heitler amplitude is well known and involves elastic form factors, which are other structure functions of the proton.

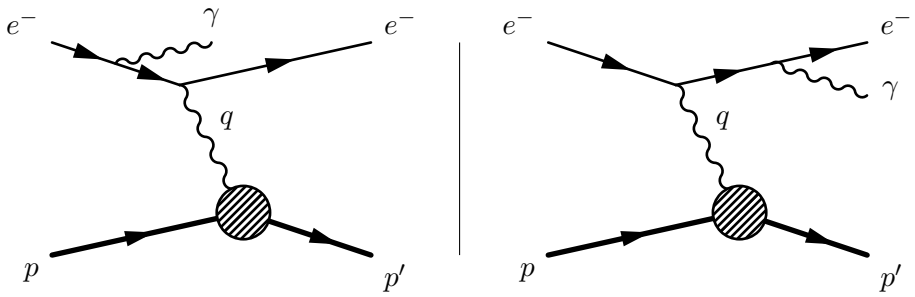


FIGURE 1.8: Feynman diagram of Bethe-Heitler process. q represents the four-momentum of the virtual photon. The real photon is either emitted by the incoming (left) or outgoing electron (right).

[5]The cross-section of a process is a function of the square of the amplitude.

DVCS and Bethe-Heitler are experimentally indistinguishable. The measured cross-section of the observed process $ep \to ep\gamma$ involves both the DVCS and the Bethe-Heitler (BH) amplitudes, therefore an interference term $\mathcal{I}$ appears:

$$\frac{d\sigma}{d\Omega} \propto \left|\mathcal{T}^{DVCS} + \mathcal{T}^{BH}\right|^2 = \left|\mathcal{T}^{DVCS}\right|^2 + \left|\mathcal{T}^{BH}\right|^2 + \underbrace{\mathcal{T}^{DVCS^*}\mathcal{T}^{BH} + \mathcal{T}^{DVCS}\mathcal{T}^{BH^*}}_{\mathcal{I}}. \tag{1.7}$$

While the square DVCS amplitude only permits to access the moduli of the CFF, the interference term $\mathcal{I}$ gives access to their real and imaginary parts since it involves the simple DVCS amplitude. In practice, both DVCS and Bethe-Heitler amplitudes as well as the interference can be expressed as a harmonic expansion of ϕ to isolate different terms.

In addition, playing with the polarizations of the beam and the target allows to measure additional observables. The interference term is notably sensitive to the beam and target polarizations. For instance, changing the helicity λ of the electron beam ($\pm\frac{1}{2}$) allows the measurement of the beam asymmetry $\mathcal{A}$:

$$\mathcal{A} = \frac{d\sigma^+ - d\sigma^-}{d\sigma^+ + d\sigma^-} \propto \frac{sin(\phi)}{d\sigma^+ + d\sigma^-} Im\left(F_1\mathcal{H} + \xi(F_1 + F_2)\tilde{\mathcal{H}} + \frac{t}{4M_p^2}F_2\mathcal{E}\right) \tag{1.8}$$

with $+$ and $-$ denoting the beam polarization and F_1 and F_2 the Pauli and Dirac elastic form factors (other structure functions of the proton that are experimentally studied since the 1950s and well known) [Guidal et al., 2013]. The latter are real numbers, thus the asymmetry gives access to the imaginary part of the CFF, which are directly values of the GPD at $x = \xi$ as seen above.

In this thesis, the final objective is to measure the DVCS/Bethe-Heitler beam spin asymmetry as a function of ϕ over the (Q^2, xB, t) phase space accessible at CLAS12 (see section 1.4 for an introduction to the practical problem and chapter 9 for asymmetry computation after event selection).

1.2.4 Accessing GPD through π^0 electroproduction

Contrary to DVCS, here a π^0 meson is produced instead of a photon (see Feynman diagram on Figure 1.9). A π^0 is not stable and rapidly decays into two photons that can be detected.

There is no equivalent of the Bethe-Heitler process for π^0 electroproduction. Therefore, this process does not have any interference and the square amplitude can be directly measured. However, there are two levels of factorization: one due to the proton structure (involving the GPD) and another one linked to the meson structure, described by the *distribution amplitude* (DA). It is therefore harder to separate the GPD from the DA contribution: instead of single convolutions like CFF, the amplitude involves double convolutions with an additional unknown hadronic structure (the DA).

However, studying π^0 electroproduction permits to access different GPD than with DVCS: it gives a favored access to the tilded GPD $\tilde{H}$ and $\tilde{E}$, and the quark transversity GPD (H_T, E_T, $\tilde{H}_T$ and $\tilde{E}_T$) are available through π^0 electroproduction but cancelled in the DVCS cross-section [Goloskokov and Kroll, 2011]. Finally, DVMP in general permits to know more precisely which quark flavor was involved in the process thanks to the knowledge of the produced meson.

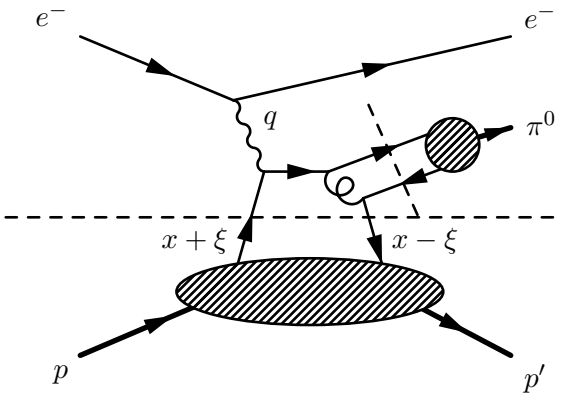


FIGURE 1.9: Feynman diagram of π^0 electroproduction. q is the four-momentum of the virtual photon. Two factorizations appear, materialized by the dashed lines: the horizontal line corresponds to the target proton while the oblique line refers to the creation of the π^0 meson.

1.2.5 Experimental status

In the remainder of this thesis, DVCS will stand for the superposition of DVCS and Bethe-Heitler since the two processes are undistinguishable. Experimentally, only a small phase space is accessible at a time and multiple facilities are needed to cover a larger phase space in Q^2, x_B. Figure 1.10 illustrates the phase space regions in (Q^2, x_B) covered by different facilities having studied or currently studying DVCS. HERMES completed their experimental program, while COMPASS and JLab 12 GeV analyses are still ongoing.

- HERMES, located at DESY laboratory in Hamburg, Germany ended in 2007. It has measured DVCS asymmetry in the intermediate values of x_B with almost every beam and target polarization combinations;
- JLab 6 GeV included the Hall A and CLAS (CEBAF Large Acceptance Spectrometer) collaborations, located at the Jefferson Laboratory (JLab) in the US. They measured DVCS in the valence quark region ($x_B > 0.1$). Since then the CEBAF accelerator upgraded to a 12 GeV electron beam to reach higher x_B and Q^2;
- JLab 12 GeV is the upgraded version of JLab 6 GeV and the experiments are still running: the objective is to study regions of the phase space with higher Q^2 (up to $\simeq 10\,\mathrm{GeV}^2/\mathrm{c}^4$) in Halls A, B and C (the CLAS12 collaboration exploits Hall B);
- The COMPASS collaboration has finished taking data in the intermediate x_B region to study sea quarks and gluons with a 160 GeV muon beam;
- In the future, the new Electron Ion Collider (EIC) [Accardi et al., 2016] (not represented on Figure 1.10) aims at reaching the very small x_B region (up to $x_B \simeq 10^{-4}$) to hit the gluon GPD.

Next section presents the CLAS12 experiment in more details as the analysis of its data is the focus of this thesis.

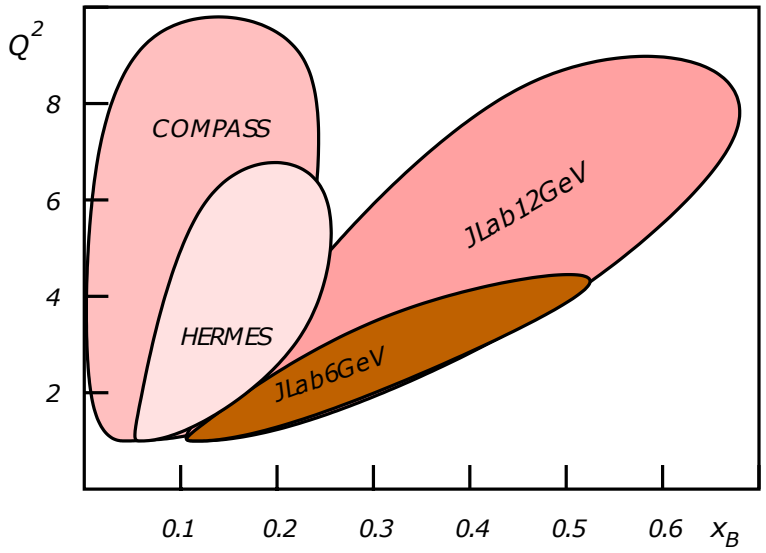


FIGURE 1.10: Phase space in (Q^2, x_B) studied by different facilities. Figure taken from [Guidal et al., 2013].

1.3 CLAS12 experiment

The Jefferson Laboratory is located in Newport News, Virginia, USA. It carries fundamental research on nucleus and nucleon structure. It comprises the CEBAF (Continuous Electron Beam Accelerator Facility) composed of two linear accelerators connected to each other with recirculating arcs. An electron beam is going up to 5.5 times through the linear accelerators to reach a maximal energy of 11.5 GeV before being sent into 4 experimental halls, from A to D (see Figure 1.11). The beam can be longitudinally polarized up to 85%.

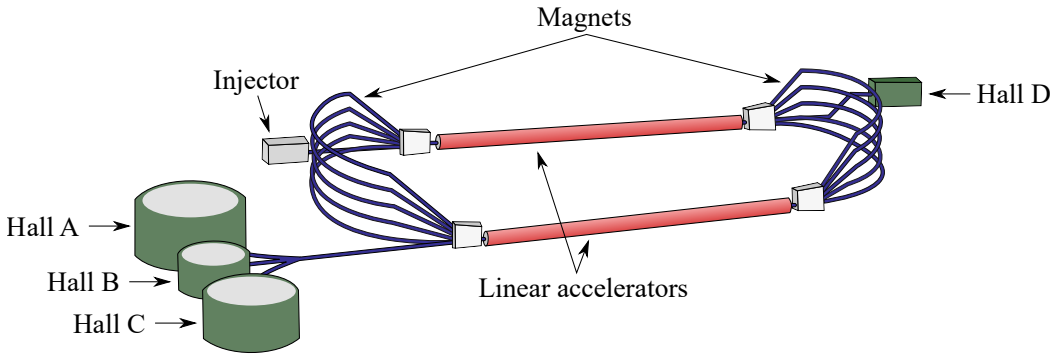


FIGURE 1.11: CEBAF accelerator and four halls.

In Hall B, the electron beam is sent onto a liquid hydrogen target, leading to a luminosity (i.e. number of events per second and per square centimeter) of $10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ for the CLAS12 experiment. Surrounding the target, the CLAS12 apparatus involves several particle detectors to characterize and identify the output particles from the collisions. The objectives of the CLAS12 collaboration are numerous and include the study of the proton structure: the electrons of the beam collide with the protons of the target. Among the possible interactions (inelastic, meson production, etc.), the objective of this thesis is to isolate DVCS events by detecting the three particles of the

final state (scattered electron, recoil proton and emitted photon) to finally measure physics observables such as cross-sections or asymmetries.

The different components of CLAS12 are described in the following subsection.

1.3.1 Experimental setup

Around the target in Hall B are three different detector packages corresponding to three different angular coverages (see Figure 1.12):

- The central detector, which aims at detecting particles with a polar angle θ with respect to the beam axis between 35 and 125 degrees. Most of the recoil protons of DVCS events are detected in this central detector.
- The forward detector, which detects particles with $5 \leq \theta \leq 40$ degrees, namely the scattered electron and the photon for DVCS and exclusive π^0 electroproduction events.
- The forward tagger to detect electrons and photons that have a small θ angle, between 2.5 and 4.5 degrees.

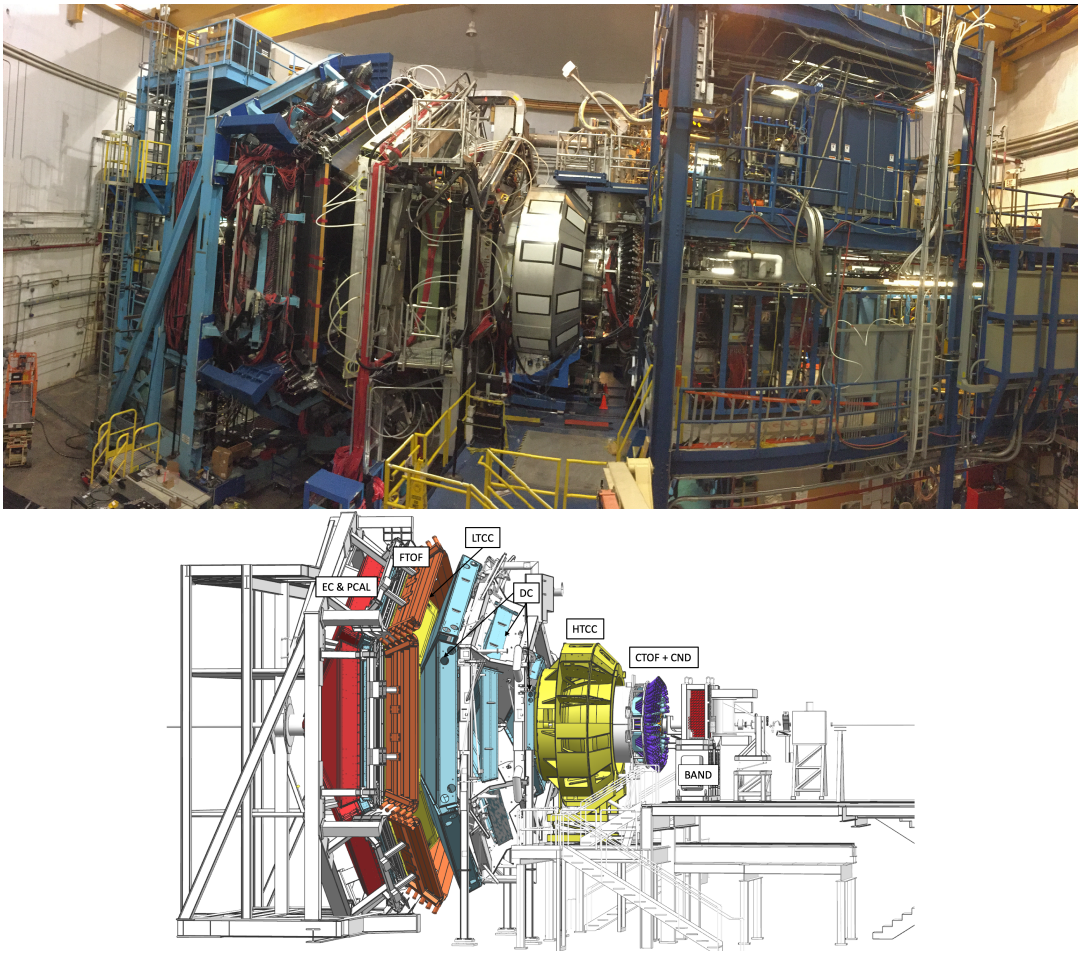


FIGURE 1.12: The CLAS12 detectors: photograph (top) and schema (bottom). The beamline comes from the right. Some components are not indicated on the schema since they are hidden by others, notably the central tracking system and the forward tagger. Both images are taken from [Burkert et al., 2020].

The different submodules of the central detectors, forward detectors and forward tagger are detailed hereafter. More details on the experimental setup can be found in [Burkert et al., 2020].

1.3.1.1 Central detector

The central detector has a cylindrical shape and is placed around the collision site. It covers polar angles from 35 to 125 degrees with respect to the beamline. It is composed of three parts: a tracking system, a time-of-flight detector and a neutron detector. It is inserted in a solenoid magnet that generates a 5 Tesla magnetic field in the direction of the beam to curve the trajectories of charged particles. The radius of curvature of the charged particles is then proportional to its transverse momentum. The measurement of this radius in tracking systems is therefore of high interest.

Tracking system

The tracking system in the central detector is composed of two detectors: a silicon vertex tracker (SVT) placed inside a barrel Micromegas tracker (BMT). Overall, the tracking system aims at detecting charged particle tracks. From recorded hits, a reconstruction software is able to compute the momentum and vertex of the particle with a certain relative momentum resolution $\frac{\Delta p}{p}$, estimated to be below 5% thanks to the combination of the two detectors. For DVCS, the central detector will mainly detect the recoil proton.

Time-of-flight detector

The objective of this detector is to identify the particles crossing it. The central time-of-flight (CTOF) detector measures the time between the activation of this scintillator and the moment the electron beam crossed the proton target. This time depends on the momentum of the particle and on its mass. The CTOF permits to distinguish π^+ from K^+ mesons up to $\approx 0.5\,\mathrm{GeV/c}$ and protons from π^+ up to $\approx 1.3\,\mathrm{GeV/c}$ (see Figure 1.13).

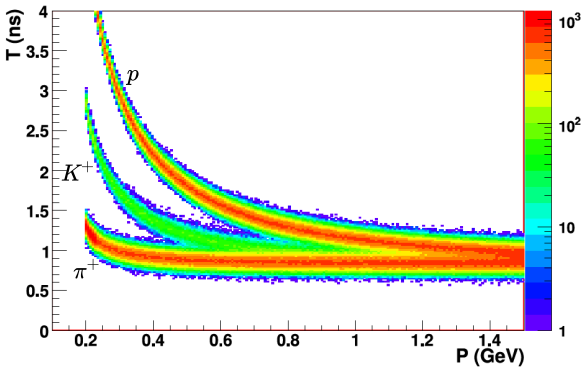


FIGURE 1.13: Time of flight of protons, positive kaons and positive pions as measured by the CTOF as a function of their momentum. Figure from [Charles, 2013].

1.3.1.2 Forward detector

The forward detector covers polar angles from 5 to 40 degrees with respect to the beamline. Ordered by increasing distance from the target, the detector is composed of:

- a high threshold Cherenkov counter (HTCC);
- a tracking system;
- a low threshold Cherenkov counter (LTCC);
- a forward time-of-flight (FTOF);
- an electromagnetic calorimeter in two parts (PCAL and EC).

A torus magnet is included in between the drift chambers. Its six coils generate a toroidal magnetic field up to 3.6 Tesla.

Drift chambers (tracking system)

Drift chambers permit to obtain a resolution up to 250 – 350 μm on the position of the particle in the detector. In the forward detector, three successive regions of drift chambers constitute the tracking system. Figure 1.14 illustrates the relative resolution expectations after reconstruction. Overall, the relative resolution $\frac{\Delta p}{p}$ is worth 0.5 – 1.5 %.

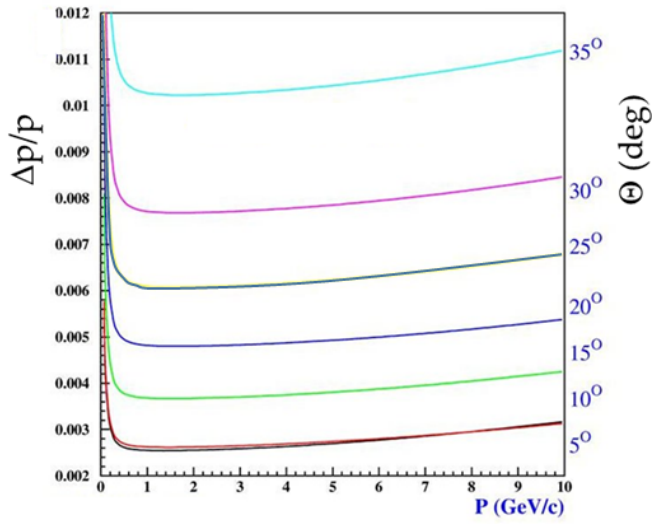


FIGURE 1.14: Estimation of the relative momentum resolution using the drift chambers at different polar angles [Mestayer et al., 2020].

Cherenkov counters

Cherenkov counters are used to improve particle identification. The detector is filled with a gas of a chosen refractive index, and a particle crossing it will generate light, a Cherenkov radiation, if its speed exceeds the speed of light in this medium.

Two Cherenkov detectors are used in CLAS12: the high and low threshold Cherenkov counters (HTCC and LTCC). They use distinct gases with different refractive indexes. The former separates electrons from pions, kaons and protons at momenta up to

4.9 GeV/c, while the latter separates pions from kaons and protons at momenta above 3.5 GeV/c.

To improve particle identification at high momentum (3 – 8 GeV/c), one sector of the LTCC has been replaced with a ring imaging Cherenkov detector (RICH). The principle is to measure the angle at which the Cherenkov light is emitted with respect to the particle's trajectory. This angle is related to the velocity of the particle. Together with the information of the momentum, the mass can be computed and therefore the particle is identified.

Time-of-flight detector

The forward time-of-flight (FTOF) has a similar functioning than the CTOF. It is designed to separate protons from kaons and pions at larger momentum values.

Calorimeters

The electromagnetic calorimeters are the outer-most detectors of CLAS12. They consist in two parts: a preshower calorimeter (PCAL) and an electromagnetic calorimeter (EC). The objective of the PCAL is to increase the granularity of the global calorimeter and notably to be able to separate two close photons, which is of high interest to reconstruct π^0 mesons. The PCAL and EC are disposed in six sectors around the beamline, at six meters of the target.

Calorimeters aim at measuring the energy of particles. They also separate electromagnetic (electron, photon) from hadronic (proton, neutron) particles by looking at the shape of the produced showers. The expected energy resolution is $\frac{\Delta E}{E} \simeq \frac{0.1}{\sqrt{E\ (\mathrm{GeV})}}$.

1.3.1.3 Forward tagger

The forward tagger is composed of a tracking system, a hodoscope to separate electrons from photons, and a calorimeter (FTCAL). Contrary to PCAL that measures only a fraction of the particle's energy, the FTCAL measures its full energy. The expected resolution is $\frac{\Delta E}{E} \simeq \frac{0.02}{\sqrt{E\ (\mathrm{GeV})}}$.

1.3.2 Accessible phase space and DVCS event detection

Because of covered angles, particle identification challenges, etc., the CLAS12 setup constrains the observable phase space. In addition, some constraints must be enforced to be consistent with theoretical hypotheses made to extract CFF from DVCS observables. Notably, the electron must have an energy over 0.8 GeV to be detected, preferably in the forward detector. Globally, the following constraints define the accessible phase space at CLAS12:

$$E' \geq 0.8\,\mathrm{GeV}, \tag{1.9}$$

$$5^\circ \leq \theta_e \leq 40^\circ, \tag{1.10}$$

$$Q^2 \geq 1.5\,\mathrm{GeV}^2/\mathrm{c}^4, \tag{1.11}$$

$$W^2 = (p_p + q)^2 = M_p^2 + Q^2\left(\frac{1}{x_B} - 1\right) \geq 4\,\mathrm{GeV}^2/\mathrm{c}^4. \tag{1.12}$$

Figure 1.15 illustrates the accessible phase space at three different beam energies.

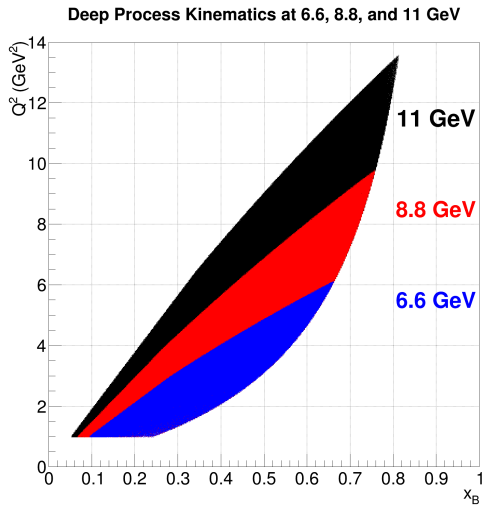


FIGURE 1.15: The (Q^2, x_B) phase space accessible at CLAS12, with three different beam energies.

The final state of a DVCS event (namely the four-momentum of the three output particles) can be computed given a quadruplet of kinematic variables (Q^2, x_B, t, ϕ). Figure 1.16 shows simulations of DVCS events following the cross-section distribution over the phase space:

- the scattered electron is emitted in majority at angles below 40 – 45 degrees, namely in the forward detector and forward tagger;
- the recoil proton has a smaller momentum ($\approx 1\,\mathrm{GeV/c}$) sent at angles most of the time covered by the central detector;
- the photon is emitted at high energies and small polar angles, it will therefore be found preferably in the forward detector or in the forward tagger.

In this way, CLAS12 earned its name of "large acceptance spectrometer", since it is designed to cover the vast majority of the theoretically accessible phase space. Figure 1.17 shows an example of a DVCS event as seen by the CLAS12 detector.

1.4 Methodology for DVCS analysis

As a summary of this chapter so far, the final objective is to learn more about the proton structure thanks to the GPD, accessed through the CFF. The CFF are themselves parameters of the DVCS cross-section and asymmetries, which are the observables to measure. The CLAS12 experiment has been designed to record DVCS events in a large phase space. The DVCS cross-section is directly proportional to the number of observed DVCS events in CLAS12, the other factors being the luminosity and CLAS12 acceptance among others. The experimental goal is therefore to isolate DVCS events from all CLAS12 data.

Two types of errors arise from this selection:

- the measurement of the cross-section has an associated *statistical error* that evolves in square root of the number of selected events;

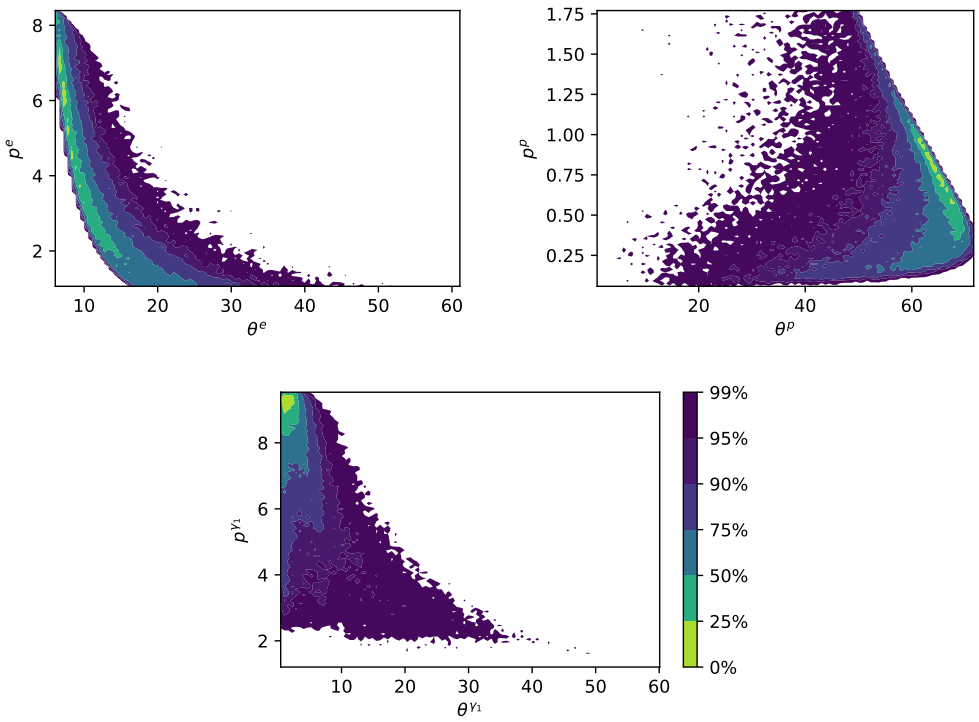


FIGURE 1.16: Simulations of the distributions of momentum and polar angle of the three particles (top left: electron; top right: proton; bottom: photon) of the output state for DVCS events, taking into account the cross-sections.

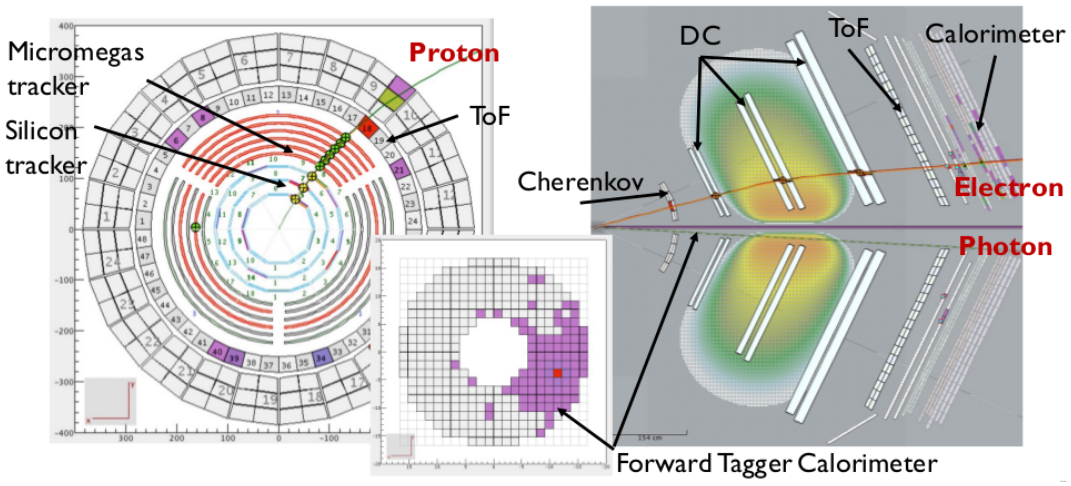


FIGURE 1.17: A DVCS event visualized in the CLAS12 event display interface. The proton has been reconstructed in the central detector, the electron in the forward detector, and the photon in the forward tagger. Credits: Maxime Defurne.

- the *systematic error* comes from all parameters involved in the measurement, notably the uncertainty on the luminosity, beam polarization, background contamination estimation, selection efficiency, etc.

Both errors contribute to the relative error associated to the measured observable (cross-section, asymmetry) and must be minimized.

In the context of DVCS analysis, the main contamination comes from π^0 electroproduction. While the DVCS final state consists of an electron, a proton and a photon, the π^0 electroproduction process final state involves a π^0 instead of the photon that decays into two photons: $ep \to ep\pi^0 \to ep\gamma\gamma$. Two cases may therefore compromise the DVCS selection:

- a DVCS event that has an additional photon for various reasons (radiation of the electron for instance) may be confounded with a π^0 electroproduction event;
- a π^0 electroproduction event in which one of the two photons has a very high energy and the other one very low may be taken for a DVCS event, because the lowest energetic photon will probably remain undetected.

Therefore, a physics analysis necessarily involves a careful event selection to retrieve the maximum of DVCS events while minimizing the π^0 contamination.

1.4.1 Standard physics analysis: event selection, π^0 subtraction and asymmetry computation

1.4.1.1 Event selection

Event selection is made based on reconstructed particle information and notably their four-momentum $p = (E, p_x, p_y, p_z)$ with E their energy in GeV and p_i the i^{th}-component of their three-momentum $\mathbf{p}$. For particles that have a non-zero mass m, the momentum $\mathbf{p}$ is proportional to the product of the mass with the velocity of the particle. Globally, the three-momentum, mass and energy of a particle are linked by the following formula:

$$E^2 = m^2 + \|\mathbf{p}\|^2 \tag{1.13}$$

where E, m and $\mathbf{p}$ are expressed in natural units, i.e. setting fundamental constants to 1. The *Minkowski norm* of the four-momentum therefore equals the mass of the particle:

$$\|p\|^2 = E^2 - p_x^2 - p_y^2 - p_z^2 = m^2. \tag{1.14}$$

During a physical interaction, all components of the four-momenta must be conserved.

Therefore, event selection is usually based on exclusivity cuts: high-level variables checking conservations of energy and momentum are examined. For instance:

- The square missing mass $ep \to ep\gamma X$ should be around 0 for a DVCS event (no other particle has been created):

$$M^2_{ep\to ep\gamma X} = \|p_{e,in} + p_{p,in} - p_{e,out} - p_{p,out} - p_{\gamma,out}\|^2 \tag{1.15}$$

  with $p_{e,in}$, $p_{p,in}$, $p_{e,out}$, $p_{p,out}$, $p_{\gamma,out}$ the four-momentum of respectively the incoming electron, proton and outgoing electron, proton and photon.

- The missing mass $ep \to e\gamma X$ should equal the mass of the proton (particle that is not counted in the outgoing particles in this missing mass), namely $0.938\,\text{GeV/c}^2$.

- The cone angle, i.e. the angle between the detected photon and the particle defined by $p_{e,in} + p_{p,in} - p_{e,out} - p_{p,out}$ (the theoretical photon) should be small.

- Etc.

Considering physics phenomena and because of the resolutions of the detectors, the distributions of these variables have a certain width around their expected value. Figure 1.18 shows the missing mass $ep \to e\gamma X$ in CLAS12 data for DVCS and π^0 electroproduction events. The cut on this variable must be optimized to keep the maximum of DVCS events while rejecting the maximum of the π^0 contamination.

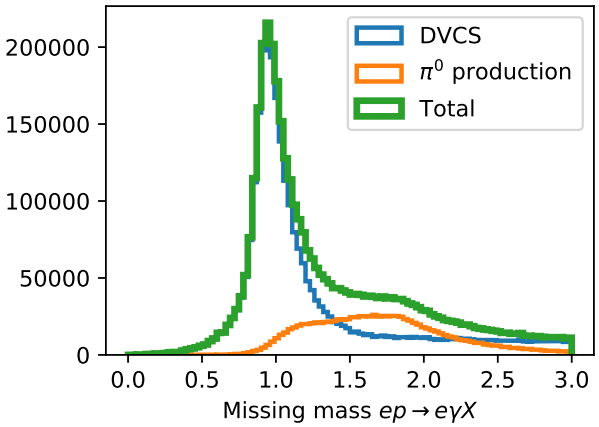


FIGURE 1.18: Missing mass $ep \to e\gamma X$ in CLAS12 simulation. The two processes have been weighted to approximately reproduce the real proportions in data.

However, part of the π^0 contamination is physically impossible to remove. Indeed, the π^0 created by the interaction between the virtual photon and the proton disintegrates into two real photons. In the π^0 rest frame, these two photons are emitted back to back in a completely symmetrical way. Their momentum in the laboratory frame is boosted by the momentum of the π^0. As a result, the two photons can have the same momentum or on the contrary have completely unbalanced momenta depending on the angle at which they were emitted (see Figure 1.19 for an illustration).

When one of the two photons has a very large energy compared to the other, it is comparable to a DVCS photon. On the contrary, the other photon may not be energetic enough to be detected in CLAS12. In this case, even the best event selection technique is not able to distinguish this kind of event from a true DVCS event given the resolutions of the CLAS12 detectors.

1.4.1.2 π^0 subtraction and asymmetry computation

An event selection method labels selected CLAS12 events as "DVCS candidates". However, amongst these candidates are a certain number of true DVCS events, but also contaminating π^0 electroproduction events and a number of various other contaminating events. The latter can be reduced to zero, however a residual π^0 electroproduction contamination remains (see the explanations above and notably Figure 1.19). This residual contamination accounts for the statistical and systematic errors. However, it is possible to estimate and subtract it afterwards. The procedure will be detailed in the last chapter together with the analysis (chapter 9).

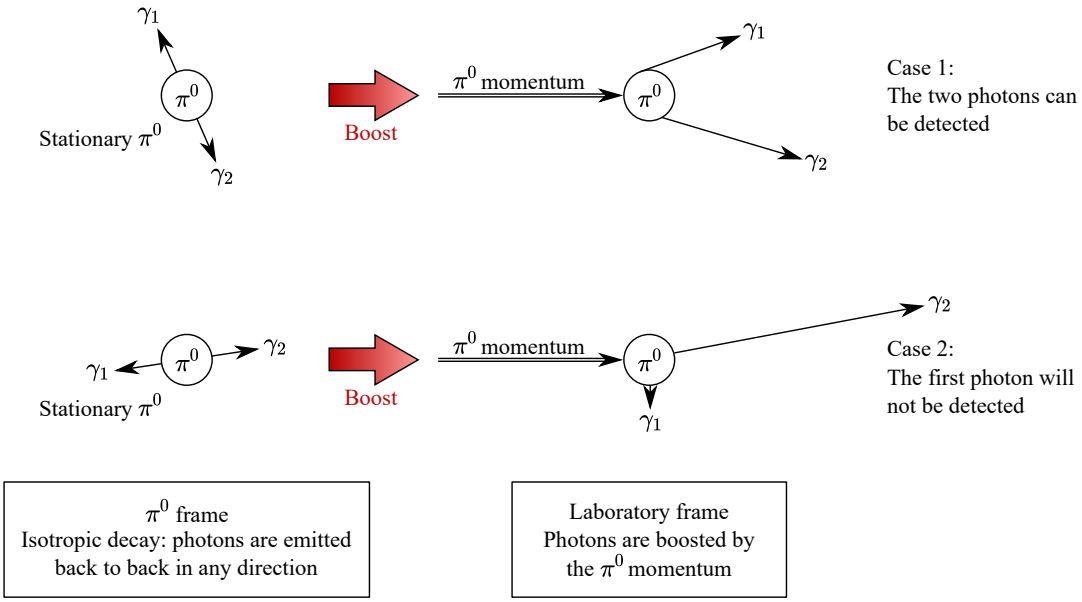


FIGURE 1.19: π^0 decay as seen in its own frame (left) and in the laboratory frame (right). In some cases, one of the two emitted photons has an energy too low to be detected (bottom case).

From the contaminated set of DVCS events selected from data and the set of identified contaminating π^0 events, the π^0 contamination can be subtracted from any distribution of the contaminated set, for instance the missing mass of Figure 1.18.

Notably, the number of contaminating events can be subtracted from the number of selected candidate DVCS events in every bin in ϕ to compute the ϕ-dependent asymmetry. The measured beam asymmetry is expressed as:

$$\mathcal{A}(\phi) = \frac{1}{P}\frac{N^+(\phi) - N^-(\phi)}{N^+(\phi) + N^-(\phi)} \tag{1.16}$$

with P the polarization degree (between 0 and 1), N^+ the number of selected events at helicity 1 and N^- the number of events at helicity -1.

1.4.2 Improving event selection with interpretable machine learning

The classic analysis method seen above is not optimized. Only linear cuts are performed on a handful of high-level variables. The values of these cuts are chosen by looking directly at the data distributions to eliminate contributions from various background processes. Moreover, the order in which these cuts are performed may not be optimized. Meanwhile, substantial investments are made in the technology, to improve the intensity of the beam or the quality of the detectors to acquire more data. However, a similar gain in statistics could be obtained by improving the analysis tools.

Besides, machine learning became a widely used tool to tackle a huge variety of problems. Its use is notably well developed in healthcare [Caruana et al., 2015], biology [Tarca et al., 2007], finance [Klaas, 2019] among others. In physics, apart from astrophysics and cosmology that have been using deep learning for several years, as reviewed in [Carleo et al., 2019], the involvement of physicists in machine learning is more recent, especially in nuclear and particle physics.

Nonetheless, the high-energy physics community is increasingly including ML programs at numerous stages: in trigger systems [Gligorov and Williams, 2013, Likhomanenko et al., 2015], generative models [de Oliveira et al., 2017], tracking (with a challenge on the Kaggle platform in 2018[6]), particle identification [Yang et al., 2005, Dery et al., 2017], or event classification [Baldi et al., 2014, Sadowski et al., 2014, de Oliveira et al., 2016]. The European Organization for Nuclear Research (CERN) developed ROOT, a library for particle physics data analysis, including TMVA (Toolkit for Multivariate Data Analysis). TMVA is a package providing tools for several ML algorithms (shallow or deep neural networks, boosted decision trees, etc.).

Machine learning has many assets to help improving event selection:

- Instead of Boolean tests in the form of one-dimensional cuts, most machine learning algorithms assign a probability to each event to be a DVCS. This permits to observe the ROC (receiver operating characteristic) curve and balance between statistics and contamination more easily [Fawcett, 2006];
- Using such algorithms should optimize the performance of event selection and retain more DVCS events with similar or lower contamination, hence diminishing the statistical and systematic errors;
- A machine learning algorithm is capable of considering detector specificities, such as the resolutions that vary geometrically or even the lower efficiency of a subpart of a detector. The selection is thus customed to the experimental setup.

Therefore, a machine learning algorithm trained on simulation data can be envisaged for event selection in CLAS12 data. As input variables, the different components of the momenta of the output particles (p_x, p_y, p_z) are available, with more elaborated high-level variables, or on the contrary lower-level detector responses.

However, the validation of an analysis method is a *sine qua none* condition for publication authorization by the physics collaboration and for acceptation at the peer-reviewing stage. The use of "black-box" machine learning is therefore hazardous for such an analysis. On the contrary, transparent machine learning models have several general advantages [Arrieta et al., 2020] that are presented in the next chapter. Additional reasons for the use of transparent machine learning for physics event selection in this thesis are:

- Optimized one-dimensional cuts (such as rule bases or decision trees) should already improve the event selection quality while being as transparent as physicists' cuts;
- The statistical and systematic errors should be at least as well estimated than with standard cuts and better estimated than with "black-box" models thanks to the ability to inspect the model in details;
- A direct control of the learned knowledge is possible. This is especially important since any supervised model needs supervised data, hence simulated data that cannot be perfectly conform to reality;
- A definitive asset is also the possibility to retro-engineer the transparent model and acquire new knowledge: the cuts usually performed on chosen variables can

[6] www.kaggle.com/c/trackml-particle-identification

be optimized, new variables relevant for event selection can be found. In addition, the analysis of the model may reveal information about the performances of the experimental setup.

However, the main negative aspect of using transparent models is their diminished performance compared to what can be obtained with "black-box" models such as deep neural networks [Došilović et al., 2018]. The comparison between the standard physics analysis, the transparent machine learning approach and the neural network approach is conducted in chapter 9. Meanwhile, the following chapter introduces interpretable machine learning in more details. After that, part II tackles the lack of performance of transparent machine learning.

Chapter 2

Interpretable machine learning

2.1 Interpretability: a general introduction 28
2.1.1 Definitions 28
2.1.2 Types of interpretability 30
2.1.3 Evaluation of interpretability 33
2.1.4 Discussion on the limits of interpretability 36
2.2 Common intrinsically interpretable machine learning models 38
2.2.1 Decision trees and fuzzy decision trees 38
2.2.2 Rule bases and fuzzy rule bases 44
2.2.3 Generalized additive models 50
2.3 Conclusion 54

Responsibility in AI has recently received an exploding gain of interest, especially in the last ten years, including aspects such as fairness, ethics, robustness, privacy, legality, etc. Two major reasons can be drawn:

- The boom of machine learning applications in many domains including healthcare [Caruana et al., 2015] or banking [Rudin, 2019] among others. These domains have a number of specific requirements to authorize the generalized use of a machine learning algorithm. Moreover, the Equal Credit Opportunity Act [Federal government of the United States, 1974] demands creditors to provide specific reasons for any decision or action taken for an applicant (e.g. credit denial or grant, account closure, etc.). Similarly, the European General Data Protection Regulation [European Parliament and Council of European Union, 2016] can enforce since 2018 a "right to explanation" concerning each algorithm that has an impact for at least one person. Notably, when a decision is made about an individual, this individual could have the right to a "meaningful explanation of the logic involved".
- The fact the machine learning models developed to tackle these diverse applications are largely based on opaque models such as deep neural networks. The exponential rise of computational resources and available data surely encouraged these developments, at the expense of transparency. The symbolic AI techniques at their peak of popularity between the 1950s and 1980s were interpretable by nature without needing to claim it.

Requesting interpretability is a popular solution to ensure compliance with the various application requirements: if the machine learning model is explainable, then its

reasoning may be analyzed to verify other criteria. Arrieta et al. [2020] give a few purposes of explainability in machine learning given the audience: managers can verify regulatory compliance, while affected users can understand their situation. Another interesting purpose they mention is for domain experts, who can trust the model itself and hopefully gain scientific knowledge. In this thesis, interpretability will indeed permit to verify the reliability of the model since specific analyses must be conducted *a posteriori* on the model when applied on specific subspaces of the phase space. Moreover, the knowledge acquired by the model will be compared to expert knowledge.

The increasing interest for explainable and interpretable AI is verifiable on Figure 2.1: the number of publications related to these fields exploded recently, especially for explainability. Moreover, Google Scholar[1] returns around 92900 results for "interpretable machine learning", including around 27300 in the last five years.

In short, this chapter aims at giving an overview of the literature in interpretability techniques. Many works already review the field of interpretability in machine learning, some of which being cited in this chapter. Section 2.1 gives definitions of several terms used in the field and introduces the next sections by describing the different types of interpretability before discussing possible methods to evaluate it and limitations of interpretability. Then, section 2.2 reviews a few popular intrinsically interpretable machine learning models.

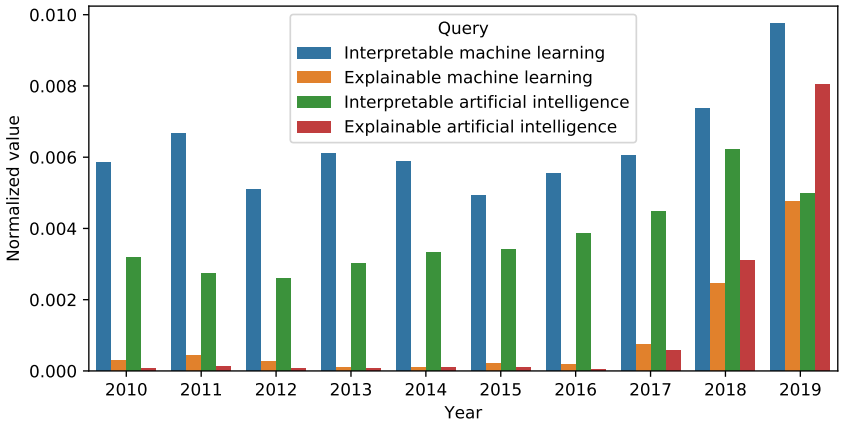


FIGURE 2.1: Evolution of the number of publications found in Scopus database[2] about interpretable or explainable machine learning and AI. On the graph, the number of publications normalized by the number of AI or machine learning related publications each year is displayed.

2.1 Interpretability: a general introduction

2.1.1 Definitions

Despite the growing interest in interpretability in the machine learning community, the consensus seems difficult on common definitions of the terms interpretability, explainability, transparency, intelligibility, etc. They are also often used in an interchangeable way.

[1] scholar.google.com (accessed 22nd April 2020)
[2] scopus.com (accessed 21st April 2020)

Several definitions for interpretability can be found in dictionaries or very recently in the AI literature:

Definition 1 (Cambridge English Dictionary[3]) *If something is* interpretable*, it is possible to find its meaning or possible to find a particular meaning in it.*

Definition 2 (Doshi-Velez and Kim [2017]) Interpretability *is the ability to explain or to present in understandable terms to a human.*

Explainability is also widely used in the literature, notably with the rise of eXplainable AI (XAI) introduced in the american Defense Advanced Research Projects Agency (DARPA) program [Gunning, 2017]. Although frequently confounded with interpretability, it has separate definitions.

Definition 3 (Cambridge English Dictionary[3]) *To* explain *is to make something clear or easy to understand by describing or giving information about it.*

Definition 4 (Guidotti et al. [2018b]) *An* explanation *is an "interface" between humans and a decision maker that is at the same time both an accurate proxy of the decision maker and comprehensible to humans.*

Definition 5 (Arrieta et al. [2020]) Explainable *Artificial Intelligence is a system that produces details or reasons to make its functioning clear or easy to understand.*

To synthesize, interpretability is often considered as a passive characteristic of the machine learning model, sometimes referred to as transparency, while explainability requires a procedure to make the model clearer [Arrieta et al., 2020]. Another crucial element in these definitions is the relativity of these notions: the interpretability of a model is always expressed in regards to a target group or individual. A layperson would need a basic explanation while an expert in the field would benefit from more complete details about the model functioning.

Doshi-Velez and Kim [2017] discuss a few dimensions of interpretability:

- *Local or global*: a model can be globally interpretable, i.e. it is understandable by itself, or only locally interpretable, i.e. an explanation can be provided for only one decision/prediction at a time.
- *Area and severity of incompleteness*: AI is generally used because of a lack of knowledge. Depending on the desired information, the explanation formulation may vary accordingly. For instance for self-driving cars, one may want to learn on what criteria they make decisions, or else check safety issues.
- *Time limitation*: in some cases, the user may need to make a rapid decision when faced with new data (e.g. a surgical operation, natural disaster, military attack, etc.). The explanation is then required to be concise and to well summarize the reasons for the model's decision. Otherwise, a deeper and exhaustive description of the model may be preferable for long-term decisions.
- *Nature of user expertise*: depending on the background knowledge of the users, the explanation may be targeted differently, as stated above.

Consequently, interpretability breaks down into several categories, described in the next subsection.

[3] `dictionary.cambridge.org/dictionary/english/interpretable` (accessed 22nd April 2020)

2.1.2 Types of interpretability

The main distinction broadly accepted in the literature is between transparent models (i.e. understanding how the model itself works) and post-hoc explanations (i.e. extract additional information from the model). A schematic illustration of this categorization and hereafter detailed subdivisions is proposed on Figure 2.2.

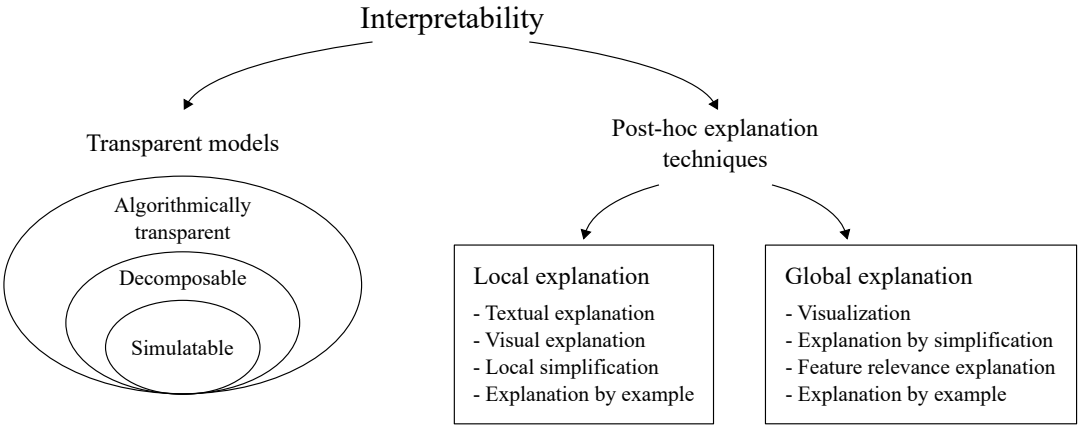


FIGURE 2.2: Overview of interpretability types (based on the classification of Lipton [2018] and Arrieta et al. [2020]).

2.1.2.1 Transparency

According to Arrieta et al. [2020], "a model is considered to be transparent if by itself it is understandable", i.e. it does not require further investigations.

Lipton [2018] and later Arrieta et al. [2020] see three different levels of transparency:

- *Simulatability*: a model is simulatable if a human is able to imitate its functioning and compute by himself its output given a set of inputs. This ability depends notably on the size of the model (a too big decision tree is not humanly simulatable) and on the amount of required computation time (a model including too many mathematical operations is harder to simulate).
- *Decomposability*: each part of a model should be explainable, including the inputs, model parameters, and computation methods. This notably implies that the input features have to be readable, i.e. anonymized and normalized features do not fulfill this requirement. The branches of a decision tree can be translated into plain sentences and understood separately, while hidden layers of a neural network need additional tools to be understood. Decomposability is also referred to as *intelligibility* in the literature [Caruana et al., 2015].
- *Algorithmic transparency*: a model is algorithmically transparent if the user can understand how the model produces the output given some inputs. While the neural network is often optimized through stochastic gradient descent, the global space being unobservable, a linear regression model is algorithmically transparent since the user can guess how the model will act given an input.

Each of these three layers of transparency contains its predecessor: a simulatable model is also decomposable and algorithmically transparent. Arrieta et al. [2020] discuss the levels of transparency of different interpretable models. References are made for each model in Section 2.2.

2.1.2.2 Post-hoc explanation techniques

Below is a classification of post-hoc explanation techniques based on [Lipton, 2018] and [Arrieta et al., 2020] and also a short review of the contributions in each category. Detailed reviews can be found in [Guidotti et al., 2018b] or [Arrieta et al., 2020].

Post-hoc explanation techniques apply to opaque models. They can be roughly divided into local and global explanation techniques: global explanations describe the behavior of the model at a global scale, while local explanations focus on subspaces of the data domain or on a single example.

Subcategories can be found below, derived from the works of Lipton [2018] and Arrieta et al. [2020] (the scope, local or global, is indicated between parentheses).

Textual explanation (local)

The principle is to generate a textual justification of a particular decision or prediction. Baaj and Poli [2019] use a natural language generator to produce example-specific explanations from a fuzzy rule base (see the review of rule bases in 2.2.2). Similarly, ExpliClas [Alonso and Bugarín, 2019] is a software capable of generating a textual explanation for the classification of an example by a decision tree or a fuzzy rule base. In computer vision, Hendricks et al. [2016] provide explanations in natural language for the classification of images using classes definitions and extraction of visual features. Dong et al. [2017] do video captioning using a long short-term memory (LSTM) to generate text.

Visual explanation (local)

Similarly to textual explanation, the objective is to explain a particular prediction, but using visualization techniques instead of natural language. This is particularly used in computer vision, notably with class-specific saliency masks [Xu et al., 2015, Zhou et al., 2016, Selvaraju et al., 2017, Fong and Vedaldi, 2017]. The principle is illustrated on Figure 2.3: a mask is extracted corresponding to the prediction. If the mask is removed from the original image, then the classifier is incapable of finding the correct class. Besides, a pointing and justification model (PJ-X) [Huk Park et al., 2018] is able to highlight the relevant parts of an image to answer a question through an attention mechanism. PJ-X also provides a textual explanation along with the annotated image. Visual explanation techniques do not exclusively apply to images: Arras et al. [2017] use a technique called layer-wise relevance propagation to highlight important words sentences for sentiment analysis.

FIGURE 2.3: Illustration of saliency masks [Fong and Vedaldi, 2017].

Local simplification (local)

Instead of giving a global explanation of how the model works on the whole data space, only a subspace of the data domain is considered for interpretation. Local interpretable model-agnostic explanations (LIME) [Ribeiro et al., 2016] is among the most known techniques for post-hoc explanation. Its principle is illustrated on Figure 2.4: it generates a local training set in the neighborhood of the data point of interest, labels these new points with the opaque machine learning model, and then trains a local surrogate linear model with the fidelity to the original model as loss function. Besides, several local explanation methods generate rules to locally explain a prediction [Ribeiro et al., 2018, Guidotti et al., 2018a].

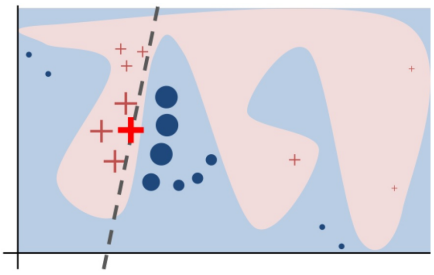


FIGURE 2.4: Illustration of the principle of LIME [Ribeiro et al., 2016].

Visualization (global)

The principle is to represent visually what the model has learned. It commonly implies dimensionality reduction, for instance using 2D visualization techniques such as t-distributed stochastic neighbor embedding (t-SNE) [Maaten and Hinton, 2008] where instances that are similarly treated by the model are represented close to each other. Some approaches in computer vision aim at visualizing the information content in the inner layers of a convolutional neural network [Mahendran and Vedaldi, 2015]. Others map the output in the input space to highlight the areas that were decisive [Zeiler et al., 2011]. Outside of the computer vision community, Cortez and Embrechts [2011, 2013] use sensitivity analysis techniques along with visualization methods to represent a model. Finally, partial dependence plots represent the relationship between one or two features and the output of the model [Goldstein et al., 2015, Krause et al., 2016].

Explanation by simplification (global)

A transparent model is trained with the objective to imitate the target model's behavior, while at the same time remaining intrinsically interpretable. This sometimes includes local simplification techniques presented above. Among global techniques, G-REX [Konig et al., 2008] is a rule extraction technique based on genetic programming that has been used as surrogate of opaque models [Johansson et al., 2004a,b]. Another example is the work of Kuralenok et al. [2019] simplifying tree ensembles through polynomials.

Feature relevance explanation (global)

The idea is to quantify the sensitivity of the model with respect to the different features. This is notably performed by Cortez and Embrechts [2011, 2013] cited above with sensitivity analysis. Game theory [Shapley, 1953] also inspired feature relevance studies: the SHAP (SHapley Additive exPlanations) method [Lundberg and

Lee, 2017] computes Shapley values of features for each prediction and summarizes feature importances and interactions globally.

Explanation by example (global or local)

Carefully chosen representative examples serve to illustrate the behavior of the model. This principle is inspired from case-based reasoning [Aamodt and Plaza, 1994]. As an example, Papernot and McDaniel [2018] use neighbors in a deep k-nearest neighbors neural network as examples to explain the prediction. Their method is used to identify biases. Kim et al. [2016] criticize the principle of using examples to explain a model: examples are insufficient to describe completely a complex data distribution, and an example-based explanation should comprise both representative examples and outliers. For instance, adversarial examples [Szegedy et al., 2013] are outliers justifying the lack of trust in such shallow explanations.

2.1.3 Evaluation of interpretability

Doshi-Velez and Kim [2017] distinguish two main cases for evaluating interpretability. Interpretability can be evaluated with respect to a specific application, and in this case the evaluation must notably take into account the target users and characteristics of the problem. Otherwise, interpretability can concern a class of models, independently of an application. To go more into details, three evaluation procedures can be drawn [Doshi-Velez and Kim, 2017]:

- *Application-grounded* evaluation: with humans and real tasks. Interpretability requirements are defined relative to the application, and evaluated with domain experts. A good baseline is human-produced explanations. A rigorous interpretability evaluation in this category is more complicated because of the difficulty to conduct a large human experiment (only domain experts are relevant).
- *Human-grounded* evaluation: with humans and simplified tasks. As the task is not directly related to a real application, a wider human experiment can be conducted to assess the interpretability of the selected model.
- *Functionally-grounded* evaluation: no humans and proxy tasks. This kind of experiments use general knowledge about interpretability of models, and the proxy is chosen accordingly. For instance, it is well known that decision trees (details in 2.2.1) are interpretable [Freitas, 2014]. Objective metrics can be used to assess the superiority of a new method over previously proposed methods along these criteria.

The remaining of this subsection presents methods of evaluation for class of models and by human-based experiments.

2.1.3.1 Class of models

Model size

One important aspect for the interpretability of given methods is their size. Explanations that are both simple and probable are indeed favored by the users [Lombrozo, 2007]. Miller's law [Miller, 1956] states that an average human can handle seven cognitive chunks simultaneously. Considering a rule base for instance, this would mean limiting the number of predicates to seven.

However, not only this limit could considerably impair the model's accuracy but is also sometimes counter-productive: Allahyari and Lavesson [2011] conducted human experiments showing a positive correlation between complexity and understandability. In their case, the datasets were small and then large models less difficult to understand. Elomaa [1994] criticizes the use of one-level decision trees and argues that medical decisions, for instance, should not rely on a single attribute. Medical experts need bigger trees they can trust to support their decisions [Lavrač, 1999]. However, Lage et al. [2019] observed an increased response time for users when faced to a model of larger complexity.

Instead of specifying a maximum size for a classification model, Freitas [2014] suggests to use instead a multi-objective approach so as to maximize accuracy while minimizing complexity.

Ranking models according to their interpretability

Freitas [2014] discusses the interpretability of different classification models through user-based experiments and draws some general conclusions: models using symbolic representations such as decision trees or rule bases (details about these models will be provided in 2.2.1 and 2.2.2) are more comprehensible to most users than mathematical equations and non-linear models. For instance, oblique decision trees (i.e. decision trees using linear combinations of attributes) are less easily understood than standard decision trees. Depending on the problem, a decision tree might be more or less interpretable compared to a rule set of similar complexity. Lakkaraju et al. [2016] found that user's understanding of the model and simulation rapidity is better when faced to rule sets (voting ensemble of IF ... THEN rules) instead of lists (ordered list of rules, i.e. IF ... THEN ... ELSE IF ... etc. rules, as explained in subsection 2.2.2). Concerning graphical models, users seem to understand better decision trees than Bayesian networks [Freitas, 2014], although no plausible explanation has been proposed.

However, Lage et al. [2019] give a few examples of studies comparing interpretability between two models, and note that the relative ordering between the two depends on the application field. In short, the choice of a model should eventually depend on the users' preferences, background, and on the characteristics of the problem. There is no total order between the machine learning models.

Finally, Freitas [2014] encourages the enforcement of monotonicity constraints, namely forcing the output of an algorithm to evolve monotonically with respect to the inputs. This constraint helps to avoid counterintuitive rules and then supports interpretability. The idea has been implemented in several studies [Fard et al., 2016a, Kotłowski and Słowiński, 2009].

Evaluating the quality of post-hoc explanations

Several general computational techniques can be used to evaluate a post-hoc explanation of a model [Gilpin et al., 2018, Mohseni et al., 2018]. They mostly rely on the objective of fidelity or model consistency with respect to the analyzed model:

- The completeness of an explanation is a measure of the distance between the predictions of the model and the explanation.

- The completeness can be measured on a substitute task depending on the explanation type: a saliency map should be compared to a direct sensitivity analysis [Mohseni et al., 2018].
- The explanation should be able to detect biases in the analyzed model.
- An explanation can be compared to those produced by other state of the art techniques or to interpretable models [Ribeiro et al., 2016].
- Edge cases should be carefully studied, when the post-hoc explanation technique gives unreliable explanations [Kindermans et al., 2019].

2.1.3.2 Application-specific human experiments

Few complete case studies on a specific task using artificial intelligence techniques also perform a rigorous human experiment to evaluate the interpretability of the deployed model. Authors of many studies simply assess that their model is "obviously interpretable" [Tjoa and Guan, 2019].

The methods used to evaluate interpretability depend on several factors. Mohseni et al. [2018] enumerate the different types of evaluation techniques depending first on the targeted users:

- The first kind of users is novice users, i.e. who are not assumed to have any prior knowledge on the problem nor on machine learning. Experiments that can be performed are studies of the mental model of these users, namely their understanding of how the model works. One drawback is that they can be biased by the format of the explanation presentation. One can also study the users' trust for the proposed model.
- Data experts have a deeper knowledge concerning the application field but do not necessarily know much about data analysis techniques. They can notably express their satisfaction for the provided explanation in several case studies.
- Machine learning experts do not have a deep knowledge about the application domain, but can conduct computational studies including checks of compliance between the interpretation and the model's accuracy.

Then, many different kinds of human experiments can be conducted. Doshi-Velez and Kim [2017] give a framework of possible experiments exclusively for simplified tasks, allowing a larger sample of users.

- Users choose between two explanations proposed for solving a problem. Lage et al. [2019] give a few examples of works using this technique, varying the explanation size, the concepts involved, and the redundancy of concepts in the explanation.
- Users have to simulate the model by making predictions manually. This permits to check that the model's functioning has been well understood. This is also called the "mental model" by Mohseni et al. [2018].
- Users are asked how to modify the model to obtain a desired output, or else how to modify the input to change the output.

Mohseni et al. [2018] give additional evaluation methods:

- Interviews and questionnaires can be used to ask users about their satisfaction about the provided explanation and about their trust in the system. Baaj and Poli [2019] for instance conduct a survey among non-expert users to evaluate generated text explanations. More precisely, they propose a number of criteria on the form and substance of the explanations.
- For domain experts, diverse performance metrics can be measured on the targeted task using the provided explanation.

Finally, Lage et al. [2019] summarize the different types of measures that can be made: the response time of the users, the accuracy of the understanding, and their subjective satisfaction.

2.1.4 Discussion on the limits of interpretability

2.1.4.1 About the notion of interpretability

Despite the urgent need of the AI community to develop more reliable models (which implies interpretability or explainability in many cases), no rigorous definition of interpretability was proposed until very recently. Whereas many papers use the terms "interpretability", "explainability", "intelligibility", "transparency", etc. in interchangeable ways, it remains to be seen whether a consensus will appear in the future.

Similarly, there exists no common rigorous framework for the evaluation of the interpretability of a system. Works that perform an in-depth study of interpretability including human experiments remain rare, and are mostly made in the context of a specific application. Doshi-Velez and Kim [2017] underline the difficulty for researchers to perform such experiments and for the community to check the rigor of their experimental design. However, such evaluations directly measure the performance of an explanation with respect to the targeted application.

2.1.4.2 On interpretability methods

Transparent models

The major argument against transparent models is their loss in performance compared to opaque models [Došilović et al., 2018]. Popular machine learning models are often represented along a curve such as the one on Figure 2.5, although the relative ordering between the models is not absolute and varies notably depending on subjective perceptions and applicative problems. Post-hoc techniques attempt to move the points to the right side of the plot.

The choice of the balance between performance and transparency depends on the application. Lipton [2018] explains that the transparency requirement for doctors to trust the system can prevent from improving healthcare in general. Emmert-Streib et al. [2020] suggest that explainability is an acceptable criterion as long as the generalization error remains within an acceptable margin.

Efforts recently appeared to merge connectionist (opaque) models with symbolic (transparent) approaches, to get the best of both worlds by trying to raise up the points on Figure 2.5 towards the top right side of the plot. Arrieta et al. [2020] give a few examples of works pursuing this idea. As an example, DeepMind propose a differentiable inductive logic programming framework to mix highly perceptive neural networks with reasoning models [Evans and Grefenstette, 2018].

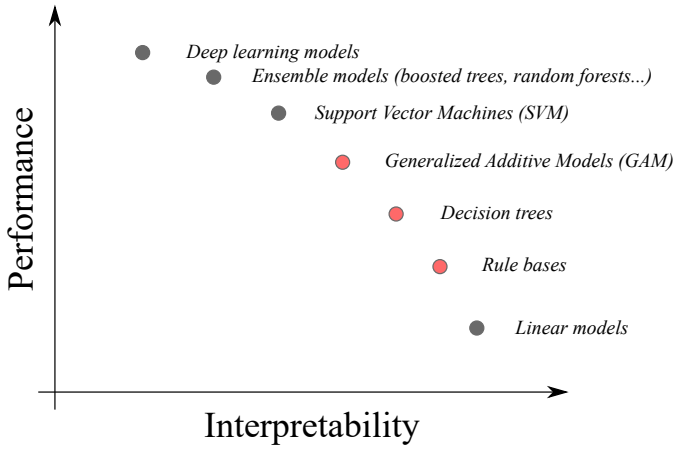


FIGURE 2.5: Representation of the relative positioning of popular machine learning models according to their perceived performance and interpretability [Arrieta et al., 2020]. The three models with red dots will be presented in 2.2).

Post-hoc explanations

As post-hoc explanations are often global simplifications or local approximations of an opaque model, they suffer from the confirmation bias. Multiple papers are questioning the validity of explanations generated from a neural network. Heo et al. [2019] show that adding a well-chosen term in the training loss permits to manipulate the interpretation computed by an external algorithm, while keeping the same level of performance. Post-hoc explanations also undergo adversarial attacks [Slack et al., 2020]: the parameters of a model can be carefully fine-tuned so that the interpretation hides some biases that are yet present within the model's functioning. Denis and Varenne [2019] underline that explanation are often biased towards persuasion instead of information. Laugel et al. [2019] point out the dangers of counterfactual examples, as they may be unjustified, namely not related to existing data.

The doubts of the machine learning community on the reliability of such techniques became important enough so that new post-hoc explanation techniques must now defend their robustness [Alvarez-Melis and Jaakkola, 2018, Melis and Jaakkola, 2018].

2.1.4.3 On the explanation complexity

It has already been mentioned that a too simple explanation could be counterproductive, both in terms of performance, user's trust and understanding [Elomaa, 1994, Lavrač, 1999, Allahyari and Lavesson, 2011]. Emmert-Streib et al. [2020] state on the contrary that explanations could be too big and complex. They draw a parallel with physics theories: explanations, here the theories such as quantum mechanics or quantum field theory, take the form of mathematical formulas that are less convenient in the context of higher level physics such as fluid mechanics or thermodynamics even if they describe completely and perfectly the system's functioning. For machine learning models explanations, they conclude that even a perfect explanation could be unacceptable: only trained experts would understand and be able to interpret the

explanation, while others (i.e. managers or end-users) would only see another opaque and non-explainable model. This confirms that the choice of the explanation should highly rely both on the targeted users and on the desired accuracy.

2.2 Common intrinsically interpretable machine learning models

As stated in the introduction of this chapter, the review proposed in this section does not aim at being exhaustive, since the number of methods considered more or less interpretable is large. The focus is on the three classes of models highlighted on Figure 2.5. Decision trees and rule-based methods are considered on the one hand. On the other hand, generalized additive models are more difficult to interpret [Arrieta et al., 2020] but are usually more performing while remaining analyzable by domain experts and notably by physicists.

Concerning the verbal models, there exist variants using *fuzzy logic* instead of Boolean logic. The advantages of fuzzy logic are numerous and are presented in Appendix A. The models themselves are presented hereafter.

2.2.1 Decision trees and fuzzy decision trees

Decision trees are machine learning models that can be used both for classification and regression. They have a tree structure in which the leaves contain the information to compute the prediction and the internal nodes are conditions on the attributes.

Building optimal binary decision trees has been proven to be a NP-complete problem [Laurent and Rivest, 1976]. Therefore, most of induction algorithms rely on heuristics such as greedy algorithms. Most of them proceed in a top-down approach by recursive partitioning:

1. an attribute A is selected regarding a measure of discrimination;
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 mostly by their splitting method and termination criteria.

2.2.1.1 Crisp decision trees

Decision trees are either vocabulary based or numerical. The latter are the only ones able to handle numerical (i.e. continuous) attributes without prior partitioning, contrary to vocabulary based trees. In the following are presented three popular tree induction algorithms for classification: ID3, producing vocabulary based trees, and C4.5 and CART, which produce numerical trees.

ID3 (Iterative Dichotomiser 3) [Quinlan, 1986] and *C4.5* [Quinlan, 2014] both use the information gain as measure of discrimination. The information gain IG of an attribute A is the difference between the entropy of the current node and the weighted

sum of the entropies of the children nodes:

$$IG = H(X) - \sum_i p_i H(X_i). \tag{2.1}$$

$$H(X) = -\sum_k p_k(X) \log_2 p_k(X). \tag{2.2}$$

X is the data contained in the current node, and the X_i are the data subsets that would be created splitting along the tested attribute A. p_i is the proportion of instances in child node i. k covers the space of data labels, and $p_k(X)$ is the proportion of class k amongst the dataset X.

Since ID3 handles only attributes that are already partitioned, a prior partitioning of numerical attributes must be conducted. This can be done through various discretization methods including for instance binning, clustering, or else the use of domain expertise.

On the opposite, C4.5 is able to handle both continuous (i.e. without prior partitioning) and discrete attributes. In addition to browsing the list of attributes, it scans each attribute in ascending order, computing the information gain associated to each candidate threshold on the attribute. The node then splits along the best attribute, producing two branches: one corresponding to the attribute values below the threshold and the other to the attribute values over the threshold. In contrast to ID3, C4.5 can reuse several times the same attribute, notably to perform several splits on the same attribute.

Once the attribute A has been selected, the node splits into several children and the algorithm resumes. A branch stops (i.e. the last node turns into a leaf) if:

- the best possible information gain is negative or null (for instance when all instances in the node belong to the same class);
- if no attribute has been found to split the node, namely the created subsets would not contain enough instances (the minimum required number is a hyperparameter of the algorithm).

C4.5 extends ID3 since it is able to handle missing values. Missing values in the training set are simply ignored during the entropy computation, while several strategies exist to deal with missing values in test data:

- instances containing missing values may be discarded;
- imputation can be conducted to make up for missing values;
- the models themselves can be adapted, for instance by adding a branch specific to missing values or by taking an approach similar to fuzzy decision trees and propagating examples with missing values in several branches simultaneously.

In addition, C4.5 performs post-pruning using a validation set: several techniques exist [Altay and Cinar, 2016]. The most common pruning technique for C4.5 is minimal error pruning: the idea is to take into account the validation error rate of a subtree with respect to the same error if it would be replaced by a leaf. A node may be replaced by a leaf, or a branch may be deleted, if the score improves on the validation set. This permits to reduce overfitting.

The *CART* (Classification and Regression Trees) [Breiman et al., 1984] algorithm can build both classification and regression trees, and uses the Gini impurity as discrimination measure:

$$I_G(X) = \sum_k p_k(X)(1 - p_k(X)) \tag{2.3}$$

with k iterating over the number of classes, and $p_k(X)$ the fraction of elements of class k in the set X. For regression, CART uses the mean squared error:

$$E(X) = \frac{1}{N} \sum_i (y_i - \bar{y})^2 \tag{2.4}$$

with N the number of instances in X, y_i the target value of example i, and $\bar{y}$ the mean of target values in set X.

CART produces only binary trees: in the case of continuous attributes it splits data along a threshold similarly to C4.5, but for discrete attributes it selects the best value against the others: one of the two created branches contains the instances taking this particular value, and the other branch gets the remaining instances. The same attribute can thus be selected more than once. CART usually performs cost complexity pruning: the pruning decision also depends on the size of the subtree, weighted by a hyperparameter α.

2.2.1.2 Fuzzy decision trees

Fuzzy extensions of ID3 have been proposed [Weber, 1992, Janikow, 1998, Wang et al., 2000]. The induction itself follows the same principle, although the computation of the information gain differs since the examples can go into several nodes at a time. Indeed, they obtain a membership degree $\mu_{X_i}(x)$ for each child node, which is combined with their current membership degree in the node. The possibilistic method [Zadeh, 1978, Dubois and Prade, 2012] uses the min operator to update membership degrees, while the probabilistic method [Tsang et al., 2009, Sébert and Poli, 2018b] progressively multiplies the degrees with the previous one. Usually, each attribute is strongly partitioned, namely $\forall x \in X, \sum_i \mu_{X_i}(x) = 1$. Then, the computation of $p_k(X)$ in the entropy takes into account these degrees:

$$p_k(X) = \frac{\sum_{i, x_i \in k} \mu_X(x_i)}{\sum_i \mu_X(x_i)}. \tag{2.5}$$

However, other discrimination measures have been employed in the fuzzy ID3 literature. Bouchon-Meunier and Marsala [2003] discuss the choice of discrimination measure for selecting an attribute in fuzzy decision trees, including information gain and an ambiguity measure [Yuan and Shaw, 1995]. Marsala [2012] proposes a discrimination measure including both the ambiguity principle of Yuan and Shaw [1995] and the graduality between the attribute and the class.

The processing of an example by a fuzzy ID3 is illustrated on Figure 2.6, with two attributes x_1 and x_2 that have been uniformly partitioned. In the end, the contributions of each leaf reached by the example are either summed (probabilistic method) or the maximum is taken (possibilistic method) to get the final prediction.

Besides, the induction of a fuzzy ID3 tree requires a prior fuzzy partitioning for the numerical attributes. While it is often designed by domain experts [Weber, 1992], a few works automate this step. Bouchon-Meunier and Marsala [1999] cite a few methods

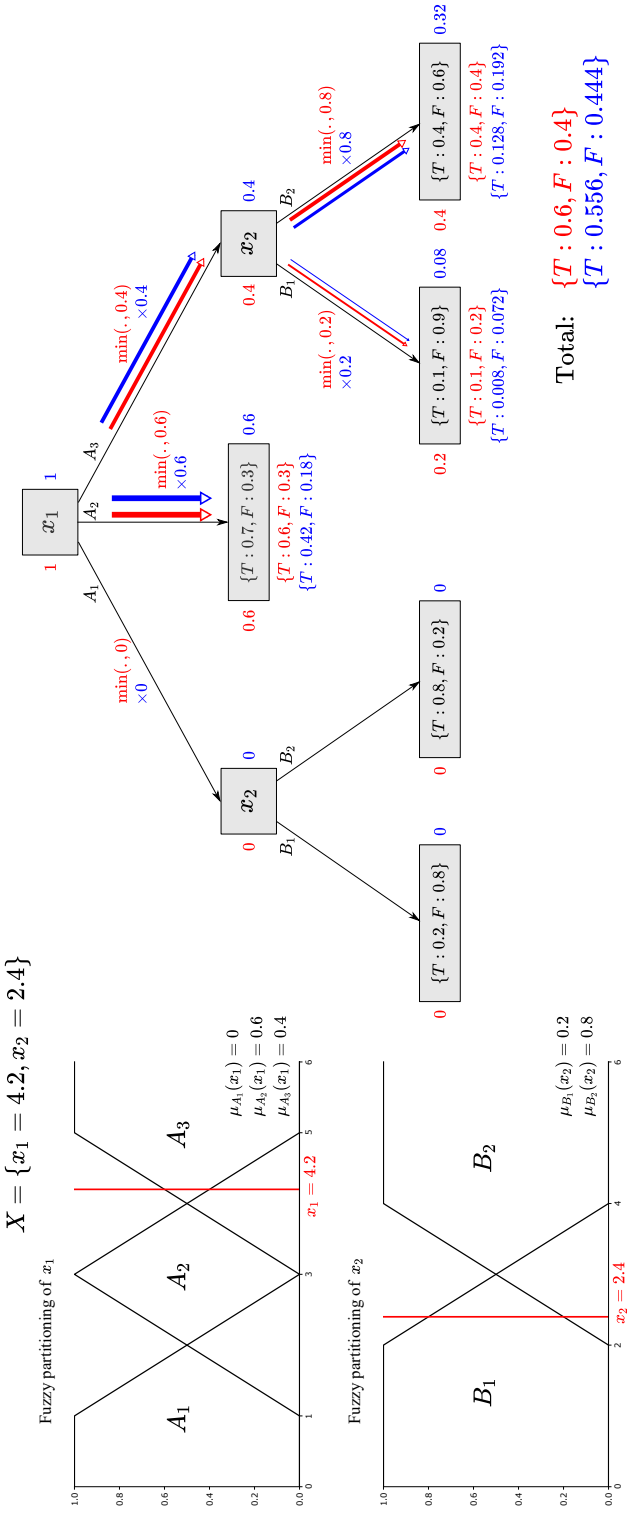


FIGURE 2.6: Example of the classification of an example X by a fuzzy decision tree with two classes T (true) or F (false). The red numbers are the membership degrees of X to the different nodes according to the possibilistic approach (min/max operators), while the blue numbers are the same degrees but following the probabilistic approach (sum/product operators).

for the determination of fuzzy modalities (another wording for determining a fuzzy partitioning). For instance, Qin and Lawry [2005] perform a uniform partitioning. Janikow [2004] pre-partitions the attributes using information gain until a maximum number of subsets is created. Pedrycz and Sosnowski [2001] use a fuzzy clustering algorithm (fuzzy C-means) to obtain a strong partition for each attribute. Sébert and Poli [2018a] perform fuzzy clustering as well with a dissimilarity measure adapted to imprecise data.

Recently, the focus in the community is on intuitionistic fuzzy decision trees based on fuzzy ID3 [Bujnowski et al., 2015], which introduce a non-membership degree in addition to the classical membership degree. The entropy computation is adapted to take into account both of these degrees.

The inference of fuzzy numerical trees is exactly the same than for fuzzy ID3: each example may be propagated through several branches simultaneously, and the final prediction consists in a weighted sum of the reached leaves.

The induction of a fuzzy numerical tree relies on the fuzzification of its thresholds. Olaru and Wehenkel [2003] first determine the optimal attribute with the corresponding threshold, and then create a linear transition between the two branches. They search an optimal width for the transition through a Fibonacci search [Ferguson, 1960] with five evaluations aiming at minimizing the root mean square (RMS) error.

Chandra and Varghese [2009] also start by finding the attribute and threshold to get the best Gini impurity, and then fuzzify the boundary with a sigmoid function using the standard deviation of the attribute:

$$\mu_R(x) = \frac{1}{1 + e^{-\sigma(x-thr)}}; \quad \mu_L(x) = 1 - \mu_R(x) \tag{2.6}$$

thr being the retained threshold, σ the standard deviation of the attribute, $\mu_R(x)$ and $\mu_L(x)$ the membership degree of x respectively to the right branch (or corresponding to $x \geq thr$), or to the left branch (or $x < thr$), as illustrated by Figure 2.7.

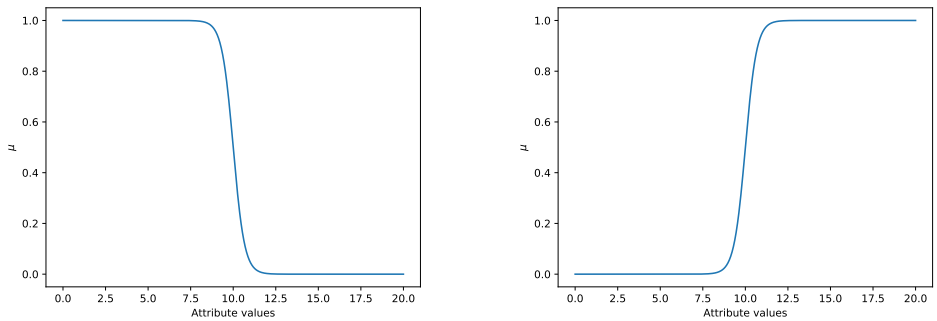


FIGURE 2.7: Examples of fuzzy membership functions to the two branches created by a split with threshold 10 and standard deviation 3: $x < 10$ on the left and $x \geq 10$ on the right.

2.2.1.3 Tree ensembles

Although they are often considered not interpretable [Arrieta et al., 2020], a few tree ensemble methods are still described here since their size can be limited and individual trees are usually smaller.

Ensemble methods combine multiple individual "weak" classifiers to obtain a better performance globally. In this case, several shallow trees are built and the global prediction is a vote amongst these individual trees. The individual trees can be induced with any algorithm, for instance ID3, C4.5 or CART.

Random forests, bagging and boosting are the most common techniques to induce tree ensembles. *Random forests* [Breiman, 2001] build a fixed number of unconstrained trees (i.e. without limitation on their size) but each tree can only use a random subsample of the input features.

Bagging methods [Breiman, 1996] select a random subsample of the training dataset to induce each tree to be built.

Finally, *boosting* methods iteratively add new trees to correct for the errors of the previous ones. Adaptive boosting (AdaBoost) [Freund and Schapire, 1995] weights the examples depending on their difficulty to be classified by the previous classifiers. The tree depth in AdaBoost is usually restricted (the depth is set to 1 in the implementation of scikit-learn[4]).

Gradient boosting [Friedman, 2001] is a subfield of boosting and combines gradient descent with boosting: the idea is to use regression trees (usually CART) as individual learners, and to update the target variable at each iteration. Given a training dataset (x, y), a current global model F_n, and a differentiable loss function $\mathcal{L}(y, F_n(x))$, the next individual tree is trained using the residuals r_n as target values:

$$r_n = -\frac{\partial \mathcal{L}(y, F_n(x))}{\partial F_n(x)}. \tag{2.7}$$

The tree model h_{n+1} is then added to the global model $F_n(x)$ with a multiplying factor γ_{n+1} chosen to minimize the loss function.

$$F_{n+1}(x) = F_n(x) + \gamma_{n+1} h_{n+1}. \tag{2.8}$$

Regularization can be applied to reduce the overfitting effect, notably through shrinkage (i.e. reducing the contribution of each individual tree), stochastic learning (i.e. using only a subsample of the data for each iteration), or limiting the individual tree size. The most popular variants of this principle are the Gradient Boosting Machine (GBM) [Friedman, 2001], eXtreme Gradient Boosting (XGBoost) [Chen and Guestrin, 2016], and LightGBM [Ke et al., 2017], the last two being optimized versions (in terms of computation and performances) of the standard GBM.

2.2.1.4 Interpretability of decision trees

According to Freitas [2014], the graphical structure of decision trees supports their interpretability: the path taken by an instance can be easily followed by a user, the hierarchy gives the relative importance of the attributes, and only a subset of the attributes is effectively used in the tree. Arrieta et al. [2020] discuss the three levels of transparency for decision trees:

- Decision trees are algorithmically transparent, since they consist in human readable rules that explain by themselves the knowledge that have been learned.

[4] scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

- Decision trees are decomposable in several rules that can be interpreted independently, provided their size is reasonable.
- Decision trees are simulatable without any background knowledge within the limits of human computation capabilities.

In short, increasing the size of a decision tree makes it progressively lose its transparency levels: reaching a certain depth, the tree will first lose its simulatability and then its decomposability. Indeed, a very deep tree is hardly decomposable since the obtained rules themselves are not simulatable.

In particular, ID3-based decision trees are considered more easily interpretable. Indeed, each value of an attribute can be named, so that the tree can be expressed in natural language. In fuzzy ID3 notably, the attributes can be linguistic variables: for instance if x represents a measure of satisfaction defined over X, X_1 can be "low satisfaction", X_2 "medium satisfaction", and X_3 "high satisfaction". Arrieta et al. [2020] agree that fuzzy versions of decision trees and rules improve understandability. However, the drawback of ID3 is that it can lead to irrelevant attributes because it creates a subtree for each value of the attributes, even if this value has no particular correlation with the target [Freitas, 2014].

Concerning tree ensembles, they are often considered opaque: post-hoc explanations are mostly used instead, mainly through model simplification and feature relevance studies [Arrieta et al., 2020]. However, since individual decision trees are usually much shallower inside an ensemble than a single classification tree, restraining the size of the ensemble could lead to a quasi-transparent model.

2.2.2 Rule bases and fuzzy rule bases

Rule bases permit to symbolically represent expert knowledge or knowledge learned from data. Decision rules are of the form IF *conditions* THEN *statement*. In machine learning, rule lists or rule sets are in majority used for classification purposes:

- In rule lists, the rules are sequentially read. The next rule is computed only if the previous one is not satisfied.
- In rule sets, all rules are computed and the final prediction is computed through a vote amongst the rules in the set. The weight of each rule notably depends on their individual accuracy.

Contrary to decision trees that partition the input domain, rule bases cover part or the totality of the input domain. Therefore, there is often a default rule in the case of rule lists, or checks for complete covering of the input domain in the case of rule sets.

While rule sets can be generated through the translation of a decision tree [Quinlan, 1987], a number of methods directly produce rule bases from training data. Fürnkranz and Kliegr [2015] propose a brief review of rule learning algorithms. The majority of rule base induction techniques relies on a sequential covering principle, while some methods rely on the extraction of association rules. Finally, population-based algorithms can be used to produce rule bases. Algorithms extracting (fuzzy) rule bases from (fuzzy) opaque models as an interpretation of the latter are not considered in the following.

2.2.2.1 Sequential covering methods

Crisp methods

The objective of covering methods is to produce a minimal set of rules that cover the training data. The principle is described in Algorithm 1 below and originally comes from Michalski [1969] with his *AQ* (algorithm quasi-optimal) algorithm. AQ performs a top-down beam search, which depends on a seed example to find the individual rules.

Algorithm 1: Sequential covering principle.

Input : X training data belonging to classes K_1,...,K_n
Output: rule set $\mathcal{R}$
Initiate an empty global rule set $\mathcal{R}$
for *each class* K_i **do**
  $P_i \leftarrow$ set of examples of class K_i (positive examples)
  $N_i \leftarrow$ set of examples of other classes (negative examples)
  Initiate an empty rule set $\mathcal{R}_i$ for class K_i
  while P_i *is not empty* **do**
    Add one rule covering some examples of P_i and no examples of N_i
    Add the new rule to the rule set $\mathcal{R}_i$ associated to K_i
    Remove the examples covered by the new rule from P_i
  Add the rule set $\mathcal{R}_i$ to the global rule set $\mathcal{R}$
return $\mathcal{R}$

CN2 [Clark and Niblett, 1989] iterates over the basic AQ algorithm by making it more robust to noise:

- the induction of a rule does not depend on a seed example: the beam search iteratively adds conditions to the rule while maximizing a criteria (for instance the purity of the covered set, the Laplace estimate, or the m-estimate of accuracy);
- the stopping criteria for the beam search is relaxed: instead of stopping when no negative example is covered like in AQ, the rule stops growing as soon as the performance of the rule (for instance the frequency of positive examples) is greater than a threshold;
- the stopping criteria of the covering algorithm is relaxed: the performance of the rule set must be above a threshold instead of waiting for all examples to be covered;
- CN2 is capable of producing unordered or ordered rules. In the former case, it applies the covering algorithm by iterating over the classes, while in the latter case CN2 determines iteratively the best conditions and then the associated class for the next rule.

In addition to pre-pruning (i.e. early stopping of the rule induction and covering algorithm), *RIPPER* (Repeated Incremental Pruning to Produce Error Reduction) [Cohen, 1995] performs post-pruning: the training set is divided into a growing set and a pruning set. It produces rule sets for minority classes first, and finally adds a default rule for the majority class. The induction of a rule is done as follows:

1. Using the growing set, grow a rule by greedily adding conditions that maximize the FOIL's information gain criterion [Quinlan, 1990]:

$$IG = p_r \left(\log_2 \left(\frac{p_r}{p_r + n_r} \right) - \log_2 \left(\frac{p}{p+n} \right) \right) \tag{2.9}$$

   with p_r and n_r the number of positive and negative examples covered by the rule, and p and n the number of positive and negative examples covered by the default rule. Cohen [1995] uses the total description length [Quinlan, 1995] as criterion to stop growing the rule.

2. Using the pruning set, prune the rule so as to maximize the performance on the pruning set. The position to cut the rule is computed to maximize the pruning metric:

$$V = \frac{p_r - n_r}{p_r + n_r}. \tag{2.10}$$

Moreover, after all the rules are inducted, RIPPER performs an optimization step. For each rule in the set, two alternative rules are produced:

- the replacement rule is grown from scratch with the objective to minimize the error on the global rule set;
- the revision rule starts from the original rule and adds antecedents also to minimize the error on the global rule set.

The rule obtaining the minimal minimum description length [Quinlan, 2014] is retained. The optimization can be repeated several times (usually twice).

Fürnkranz [2002] uses RIPPER in a round robin classification, namely they induce one model per pair of classes. Each model consists in a rule set describing the first class, a default rule being implemented for the second class. This method, called *R3* for Round Robin RIPPER, outperforms RIPPER on a number of datasets.

Finally, Janssen and Fürnkranz [2010] compare several choices of heuristics for rule induction and find indicators of a good heuristic. For instance, consistency (related to the number of covered negative examples) must be favored against coverage.

Fuzzy methods

A number of fuzzy rule learning algorithms extend the covering principle of crisp rule induction algorithms.

Fertig et al. [1999] perform a fuzzy partitioning of all continuous attributes before applying a beam search to obtain a rule. Cloete and van Zyl [2006] include this technique into a full covering algorithm.

FR3 (Fuzzy Round Robin) [Hühn and Hüllermeier, 2008] builds upon R3. It performs a post-processing of the rules to fuzzify their crisp boundaries and obtain fuzzy sets for each antecedent in the form of trapezoidal membership functions. More precisely, the algorithm searches an extension of the support of the crisp boundary for every antecedent: the bounds of the support are greedily optimized by trying each example that is covered by the rule so far, as illustrated on Figure 2.8. The boundaries that lead to the best rule purity (i.e. proportion of positive examples) are retained.

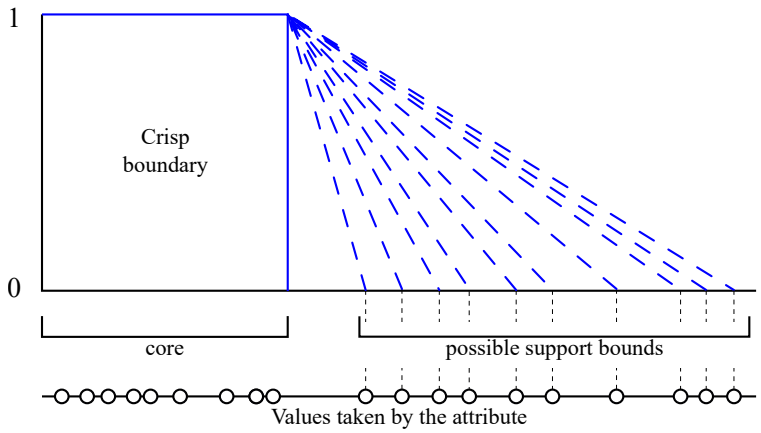


FIGURE 2.8: Fuzzification procedure for an antecedent in FR3 and FURIA.

FURIA (Fuzzy Unordered Rule Induction Algorithm) [Hühn and Hüllermeier, 2009] extends directly RIPPER with two main contributions to improve the crisp algorithm:

- FURIA produces unordered rule sets instead of decision lists, by implementing a one-vs-rest decomposition for every single class, which does not favor any default class. Rule stretching permits to classify examples that are not covered by any rule: each rule is generalized by removing the minimal number of antecedents so that the considered example gets covered. Then, the example is classified by the generalized rule, which obtains the best Laplace accuracy on the training set. This strategy has been experimentally proven to be better than just predicting the majority class.
- FURIA fuzzifies the antecedent boundaries in the same way as FR3.

The additions of FURIA obviously demand a higher computation time, both to fuzzify the antecedents and to classify uncovered examples. However, Hühn and Hüllermeier [2009] claim that FURIA obtains better performances than other fuzzy rule induction algorithms and than C4.5 on a number of problems.

2.2.2.2 Methods based on association rules

The principle of such methods is first to extract properties from data in the form of association rules, and then use them to classify new examples.

Association rules

Association rules permit to find correlations and frequent patterns in data. They are of the form "IF X THEN Y", with X and Y being two disjoint *itemsets* (i.e. statements on attributes), but Y does not necessarily correspond to a target label. An itemset can be the intersection (i.e. conjunction of premises) of several itemsets. The relevance of an association rule can be measured through two criteria:

- its support $s = \frac{\text{card}(X \cap Y)}{\text{card}(E)}$, with $\text{card}(E)$ the number of instances;
- its confidence $c = \frac{\text{card}(X \cap Y)}{\text{card}(X)}$.

Popular association rule extraction algorithms are for instance A Priori [Agrawal et al., 1994], Eclat [Zaki, 2000], and FP-Growth [Han et al., 2000]. The principle is to find all

itemsets that have a minimal support (otherwise, the itemset could be due to noise) and a minimal confidence.

The search for frequent itemsets is often exhaustive. As an example, A Priori iteratively builds itemsets of increasing size: with the frequent itemsets of size 1, it builds all possible itemsets of size 2 and keeps the ones with sufficient support, and continues until no new itemset with enough support can be built. In a second phase, which is mostly common to all association rule extraction algorithms, all possible rules are built (varying the relative position of the implication sign) from resulting itemsets. Only the rules with sufficient confidence are retained.

Besides, fuzzy versions of association rule mining algorithms have been proposed [Dubois et al., 2006, Hong and Lee, 2008, Pierrard et al., 2018]. The quantitative values are transformed into linguistic terms (namely fuzzy sets). The support and confidence measures are extended:

$$s = \frac{\sum_{z \in E} \mu_{XY}(z)}{\text{card}(E)}, \tag{2.11}$$

$$c = \frac{\sum_{z \in E} \mu_{XY}(z)}{\sum_{z \in E} \mu_{X}(z)}. \tag{2.12}$$

The search is based on these fuzzy counts using the membership functions to the different itemsets. As an example, a fuzzy version of FP-Growth has been developed [Wang et al., 2010].

Classification by association

Several rule learning algorithms use association rules to produce a classification model. One option is to simply keep the association rules that have a subset of the target variable as consequence [Liu et al., 1998]. CMAR [Li et al., 2001] optimizes the process by producing exclusively association rules that have a specific class as consequence. Sulzmann and Fürnkranz [2008] compares several techniques to produce rule sets from association rules, notably selection and combination techniques. For instance, Letham et al. [2015] create Bayesian rule lists directly through frequent itemsets obtained by the FP-Growth algorithm. Then, they learn an associated label for each pattern. The rule list is iteratively modified by adding, removing or switching rules, following a Bayesian approach in which the prior distribution permits to limit the number and length of rules.

FARC-HD [Alcala-Fdez et al., 2011] extends the crisp approach with fuzzy logic for high-dimensional problems. From fuzzy association rules generated by a fuzzy A Priori algorithm, candidate rules are preselected and a genetic algorithm is performed to select and tune the final rule set. Similarly, Antonelli et al. [2015] extend the CMAR algorithm and use the association rules generated by a fuzzy FP-Growth algorithm to generate a fuzzy rule set for classification.

2.2.2.3 Population-based algorithms

In addition to the traditional top-down induction of the individual rules, population-based algorithms are also widely used: swarm intelligence and genetic algorithms can both represent rules as individuals so as to optimize their antecedents, antecedent parameters and consequences. Fernández et al. [2010] and Martens et al. [2011] propose respectively a review of genetic algorithms and swarm intelligence algorithms applied

to data mining and particularly to rule learning. Besides, Herrera [2008] reviews genetic fuzzy systems.

Some of these rule induction methods still exploit the sequential covering framework, while adapting the rule induction technique:

- Ant-colony optimization (ACO) [Dorigo and Di Caro, 1999, Dorigo et al., 2006] is based on the biological phenomenon: ants choose a trajectory and drop a quantity of pheromones proportional to the quality of their choice. Next ants are attracted by the solutions with the most pheromones and the population converges progressively. Ant-Miner [Parpinelli et al., 2002] encodes rules into trajectories, and iteratively adds rules obtained through such an algorithm to the global rule base.
- Particle swarm optimization (PSO) [Kennedy and Eberhart, 1995] consists in a population of particles, each one being a candidate solution. Particles have a position and velocity and move in the search space to find the optimal solution. Sousa et al. [2004] encode rules into the positions of particles to optimize one rule at a time.
- Genetic algorithms (GA) evolve a population of chromosomes through mutations, crossovers, and survival of the fittest. For rule induction, SLAVE [González and Pérez, 1999] evolves a population of candidate fuzzy rules by updating their antecedents, consequences, and membership function parameters.

Hybrid methods combine several of these algorithms, such as PSO and ACO for Holden and Freitas [2008].

Other methods move away from the sequential covering idea and use boosting instead: Del Jesus et al. [2004] and Sánchez and Otero [2007] use adaptive boosting to iterate over the rule set and a GA to produce individual fuzzy rules.

Finally, other methods directly optimize the total rule set. Greene and Smith [1993] perform a cooperative GA in which several populations of rules are evolved simultaneously: each population contributes to one rule in the rule set, and individual rules must compete to remain the fittest to stay in the rule set. Ishibuchi et al. [1997] propose a two-objective GA to maximize the accuracy and minimize the number of fuzzy rules in the final fuzzy rule set. Olmo et al. [2010] create rules through ACO with a niching approach to constitute the final rule set: diversity of solutions is maintained during the ACO and the most accurate ones are retained to form a rule set. Otero and Freitas [2013, 2016] directly create an entire list of rules through ACO.

2.2.2.4 Interpretability of rule bases

The interpretability of rule bases is closely related to the one of decision trees. Compared to the graphical representation of decision trees, rules have a textual representation, which hides the relative importance of the input variables [Freitas, 2014]. However, the very nature of rule sets allows the users to decompose the set and analyze the rules one by one, which makes a rule set easier to understand than a decision tree. Converting a decision tree to a set of rules is indeed a good method to improve its understandability [Quinlan, 1987]. However, rule lists are less interpretable than rule sets: understanding rules at the end of the list is harder since their interpretation must be done in the context of all of the previous ones. On another note, rules are

less prone to having irrelevant conditions since they do not have to be mutually exclusive like the branches of a decision tree. Rule sets are usually more compact than a decision tree converted to rules [Freitas, 2014].

Concerning the transparency of rules bases, Arrieta et al. [2020] underline the strong relationship of rule base transparency with their complexity:

- If the variables in the rule are humanly comprehensible and the rule base size is manageable by a human, then the rule base is simulatable.
- When the rule sizes become too large, the rules need to be decomposed in chunks to be understandable.
- Finally, a too large rule set makes the base only algorithmically transparent: mathematical tools are needed to inspect the model in further details.

The interpretation of "too large" depends on the particular characteristics of the problem and on the appreciation of the users.

Besides, fuzzy rule bases lead to a better interpretability than crisp rule bases: they use linguistic terms and are often more concise than crisp bases thanks to fuzzy logic, which is more flexible than standard Boolean logic [Cpałka, 2017, Arrieta et al., 2020]. Triangular or trapezoidal shapes work well as the membership functions while being simple [Barua et al., 2013]. Mencar [2013] discusses the interpretability of fuzzy systems in particular and provides several interpretability criteria. Notably, he states that continuity is one of the main asset of fuzzy sets, as it better reflects our natural representation of concepts.

2.2.3 Generalized additive models

2.2.3.1 From linear models to generalized additive models

Generalized additive models (GAM) [Hastie and Tibshirani, 1986] are statistical models derived from generalized linear models and additive models.

A simple linear model can be written:

$$\widetilde{y} = \beta_0 + \sum_j \beta_j x_j \tag{2.13}$$

with $\widetilde{y}$ the prediction, x_j the input variables for example x and β_j the parameters to determine. The β_j (including β_0) are obtained by a least squares estimation. From there, generalized linear models add a link function g between the linear predictor and the model's output:

$$g(\widetilde{y}) = \beta_0 + \sum_j \beta_j x_j. \tag{2.14}$$

There exist several common link functions depending on the desired output distribution. For logistic regression for instance, $g(\widetilde{y}) = \log \frac{\widetilde{y}}{1-\widetilde{y}}$. The estimation of the β_j parameters is obtained following an iteratively reweighted least squares algorithm. Each step actualizing the β_j involves a weighted least squares estimation with the weights being a function of the residuals.

Then, additive models are of the form:

$$\widetilde{y} = \beta_0 + \sum_j f_j(x_j) \tag{2.15}$$

where the f_j are "smooth" functions, which is a rather vague term. Historically, piecewise linear functions or smoothing splines [Wood, 2003] were used as the f_j and fitted by a backfitting algorithm [Hastie and Tibshirani, 1986]:

1. β_0 is initialized to obtain a unique solution: $\beta_0 = \frac{1}{n}\sum_i y_i$.
2. The shape function f_1 is learned on the residual $y - \beta_0$.
3. The next shape functions f_{k+1} are fitted on the residuals $y - \beta_0 - \sum_{j=1}^{k} f_j(x_j)$.
4. When d shape functions have been fitted, the first one is discarded and refitted on the residuals $y - \beta_0 - \sum_{j=2}^{d} f_j(x_j)$, and so on.

Finally, this leads to GAM:

$$g(\widetilde{y}) = \beta_0 + \sum_j f_j(x_j). \tag{2.16}$$

A visualization of a three-term GAM is displayed on Figure 2.9.

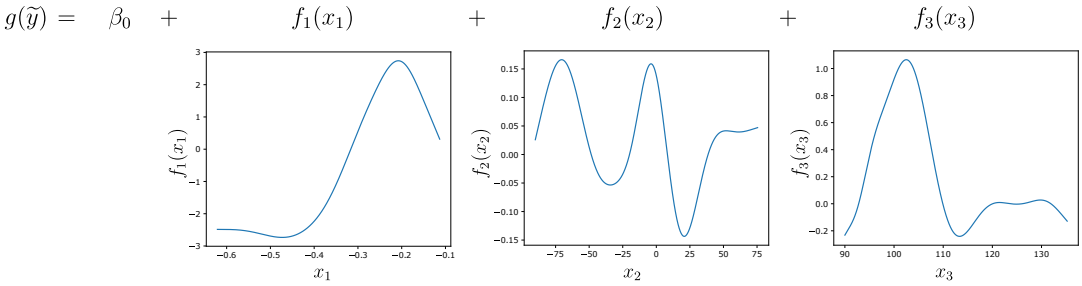


FIGURE 2.9: A GAM with three terms.

2.2.3.2 Shape functions and fitting methods

As stated in equation 2.16, GAM are a sum of smooth univariate functions. Similarly to additive models, the shape functions can be for instance splines [Wood, 2003], which are commonly written:

$$f_j(x) = \sum_k \beta_{j,k} b_k(x) \tag{2.17}$$

with b_k being basis functions (B-splines) and $\beta_{j,k}$ the parameters to fit. More recently, tree ensembles of a single variable have been used as shape functions [Lou et al., 2012]. Lou et al. [2012] find that using boosting and bagging tree ensembles as shape functions leads to better results than using splines in general cases.

Methods based on backfitting, iteratively reweighted least squares [Wood, 2017] or gradient boosting [Tutz and Binder, 2006] can be used. Below is a short description of these three fitting methods.

Backfitting

A generalization of backfitting called local scoring algorithm can be used for classification purposes (i.e. $g(\widetilde{y}) = \log \frac{\widetilde{y}}{1-\widetilde{y}}$):

1. $\hat{y} = \frac{1}{n}\sum_i y_i$. β_0 is initialized to obtain a unique solution: $\beta_0 = \log \frac{\widetilde{y}}{1-\widetilde{y}}$. All shape functions are set to 0.
2. At a given step k, let $F(x) = \beta_0 + \sum_j f_j(x_j)$ and $p(x) = \frac{1}{1+e^{-F(x)}}$.
   (a) The "residuals", i.e. target values for the current backfitting step, are $r_i = F(x_i) + \frac{y_i - p(x_i)}{p(x_i)(1-p(x_i))}$ for each training example x_i.
   (b) In contrast to the backfitting algorithm for additive models, the shape function fitting is here weighted by $w_i = p(x_i)(1-p(x_i))$.

Gradient boosting

Lou et al. [2012] use gradient boosting to learn the shape functions for all input variables. They loop over a number of iterations and over the input variables. For each input variable j in iteration m, they:

1. compute the residuals (different in case of classification or regression);
2. fit an univariate function of variable x_j on the residuals;
3. add this new function to the shape function f_j.

Iteratively reweighted least squares

In the case where the shape functions are smoothing splines, an iteratively reweighted least squares algorithm can be used. The advantage is that a penalization on the shape function complexity can be added (thus solving a penalized iteratively reweighted least squares problem). The fitting of the f_j is done by minimizing the penalized sum of squares:

$$\min_{\beta} \left\{ \left\| y - B^T\beta \right\|^2 + \lambda \beta^T P \beta \right\}, \quad \beta = (\beta_{j,k}) \tag{2.18}$$

with B the vector of $b_k(x_j)$ and $\lambda\beta^T P\beta$ a penalty term. P can for instance penalize the differences between adjacent $\beta_{j,k}$, 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) [Gu and Wahba, 1991].

2.2.3.3 Generalized additive models plus interactions

Lou et al. [2013] introduce GAM plus interactions (denoted as GA^{2}M), in which a number of bivariate terms are added to the model:

$$g(\widetilde{y}) = \beta_0 + \sum_j f_j(x_j) + \sum_{(i,j)\in\mathcal{I}} f_{i,j}(x_i, x_j) \tag{2.19}$$

where $\mathcal{I}$ is the set of feature pairs used to model pairwise interactions, allowing to exploit 2D-correlations. Lou et al. [2013] first build a standard GAM model, and then use a fast interaction detection algorithm to select the most relevant feature pairs to fit additional bivariate shape functions. They get a significant improvement over the

results obtained without pairwise terms. An example of a GA^{2}M with three univariate terms and two bivariate terms is illustrated on Figure 2.10.

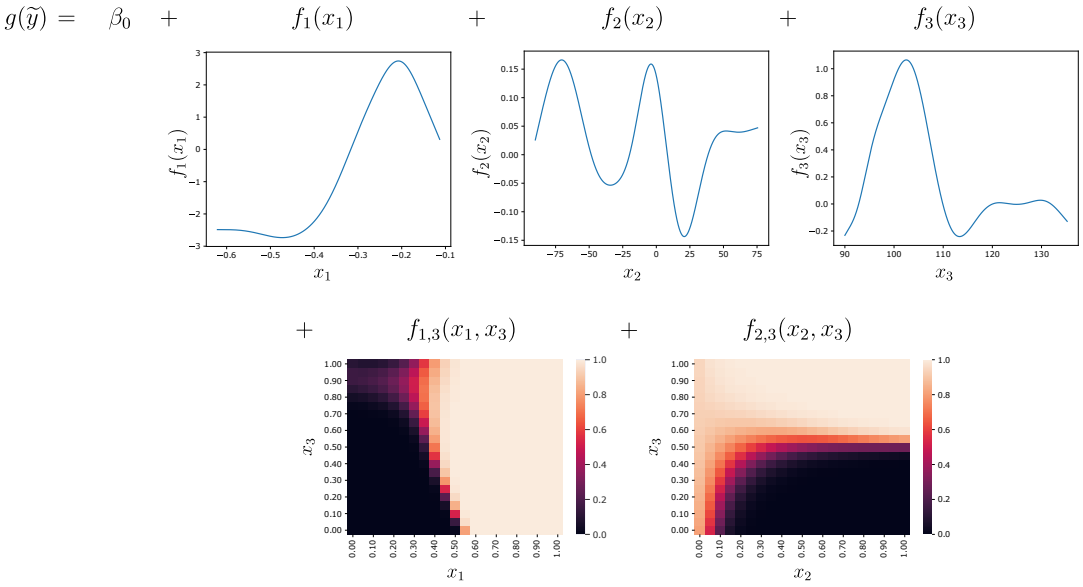


FIGURE 2.10: A GA^{2}M with three univariate terms and two bivariate terms.

2.2.3.4 Interpretability of generalized additive models

Caruana et al. [2015] claim the intelligibility of GAM and GA^{2}M since univariate and bivariate functions can be visualized (respectively as curves and heatmaps as on Figure 2.10). They find that users may prefer graphs to numbers, hence facilitating their understanding. They examine a few learned univariate and bivariate terms for pneumonia risk prediction and find them consistent with expert knowledge. However, they find that using splines leads to a degradation in performance compared to tree ensembles because of their smoothness.

Examples of the same terms learned with tree ensembles or splines are illustrated on Figure 2.11. The tree-ensemble terms are wigglier than the spline terms. However, Ross and Doshi-Velez [2018] find that regularizing gradients improves interpretability and robustness for neural networks. If extrapolated to GAM, this would mean an argument in favor of splines against tree ensembles. Objectively, the small variations in the tree-ensemble terms are difficult to interpret while they are absent in the spline terms.

According to Arrieta et al. [2020], GAM are algorithmically transparent with the help of mathematical and statistical tools to analyze them. They are decomposable by nature since each term can be interpreted independently. However, Arrieta et al. [2020] state that interactions are too complex to be simulated. The variables and smooth functions involved in the model must be "constrained within human capabilities for understanding" so that the global model is simulatable. The complexity of the link function g must also be taken into account.

Finally, GAM have been used in several applications such as healthcare [Caruana et al., 2015], finance [Calabrese et al., 2012], energy [Pierrot and Goude, 2011], environment [Murase et al., 2009], etc. for their understandability: such models are indeed often used to discover knowledge in data in addition to obtaining high accuracy. GAM are

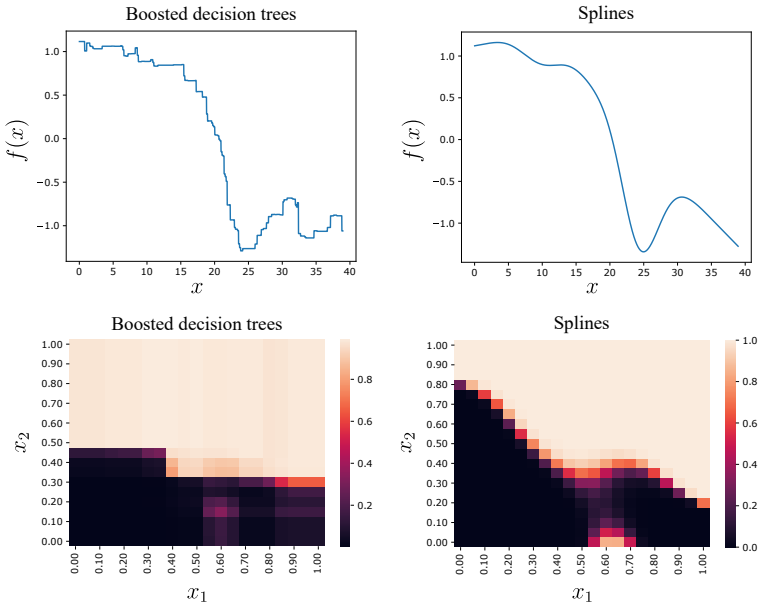


FIGURE 2.11: Comparison of two terms learned with tree ensembles (on the left) and with splines (on the right), for an univariate term (on the top) and a bivariate term (on the bottom).

considered more accurate than other transparent models such as decision trees while remaining intelligible.

2.3 Conclusion

This chapter presented the tools in interpretable machine learning that will be exploited in the remaining of the thesis. We made a summary of the challenges around the notion of interpretability in machine learning, including its definition, classification, evaluation methods and limits. We presented three main classes of transparent models, notably tree-based models, rule-based models and GAM. These three categories will serve as a basis for the developments in this thesis. One main limitation of transparent models is their diminished classification performance compared to models that are more opaque. We will particularly focus on this aspect. We will also rely on the interpretability evaluation guidelines detailed in 2.1.3 to assess the interpretability of the models we will use for the physics analyses.

In the next chapter, we precise our positioning in this thesis by preparing training datasets and baseline models.

Chapter 3

CLAS12 simulation and baselines using transparent models

3.1 Monte Carlo simulation of DVCS and π^0 production events 55
3.1.1 Simulation pipeline 55
3.1.2 Constitution of training datasets 57
3.2 Baselines using transparent models 60
3.2.1 Selected transparent models 60
3.2.2 Baselines 63
3.3 Proposed approach 64

This chapter prepares the ground for the remaining of the thesis. Since the supervised learning algorithms presented in chapter 2 need labeled data, simulations are used to constitute the training samples. Section 3.1 presents the simulation process and how to constitute a training dataset for machine learning. Then, section 3.2 lists the selected transparent models that will be used in the thesis and shows some baselines.

3.1 Monte Carlo simulation of DVCS and π^0 production events

Since the objective is to select DVCS events, two classes are used for training machine learning models: the signal class, composed of simulated DVCS events, and the background class. Since the main background to DVCS is π^0 production, the background class is only composed of simulated π^0 production events.

Data simulation is done through a Monte Carlo simulation in several steps, detailed in the following.

3.1.1 Simulation pipeline

The overall simulation and reconstruction process is illustrated on Figure 3.1.

The simulation process involves two steps:

1. *event generation*: kinematic variables (Q^2, x_B, t, ϕ) are drawn from the phase space following a certain distribution. Then, the output state of DVCS or π^0 production events is computed from these phase space coordinates;

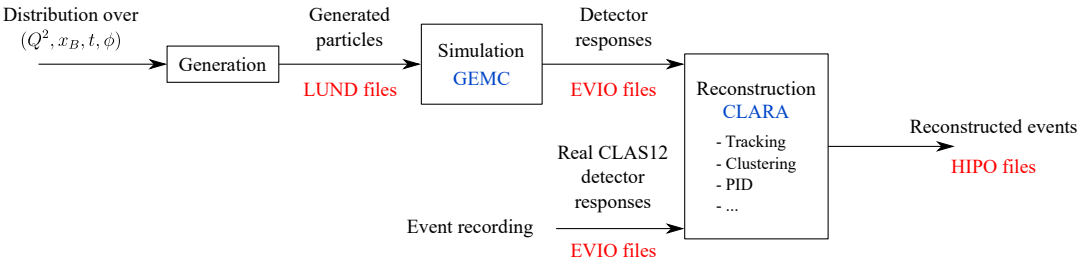


FIGURE 3.1: CLAS12 simulation and reconstruction.

2. *detector simulation*: the passage of the generated output particles through the detectors is simulated and artificial detector responses are created.

Step 1 requires the knowledge of the reachable phase space, as determined in 1.3, and the knowledge of the cross-sections to get a probability distribution for the kinematic variables (Q^2, x_B, t, ϕ). Models of DVCS and π^0 production cross-sections exist based on the data from previous experiments [Drechsel et al., 1999]. For the CLAS12 phase space, Valery Kubarovsky proposed an empirical fit of the cross-sections of π^0 production, available online[1]. The so-called VGG model of GPD is used to model DVCS cross-sections [Vanderhaeghen et al., 1999, Goeke et al., 2001, Guidal et al., 2005]. However, since DVCS data has not yet been analyzed in the CLAS12 phase space, these models are subject to potentially large errors and must be taken with caution.

From a set of picked kinematic variables, the full output state of a DVCS or π^0 production event is computed. This is the generated event, with true particle's momenta. Then, step 2 involves detector simulation. To this extent, GEANT4 is a framework for the simulation of the passage of particles through matter. Therefore, the modeled CLAS12 detectors are implemented in this framework. GEANT4 Monte Carlo (GEMC) has been developed to interface event generation with GEANT4 [Ungaro et al., 2020]: it takes as input LUND files containing information about the generated particles, and outputs detector responses under the form of EVIO files. It is flexible as the nature of the target or the magnetic fields can easily be modified.

Then, the *reconstruction* is the third step, common both to simulated and real data. Indeed, it takes detector responses as input. The principle is to group the detector signals that are associated to the same particles, reconstruct the momentum and energy of these particles and identify their nature:

- For charged particles, tracking consists in grouping the detector hits belonging to a same particle (pattern recognition) and fit its trajectory. The charge, momentum and vertex of the particle are inferred from the curvature of the track in a given magnetic field.
- Particle identification systems (Cherenkov counters, time of flight detectors) differentiate several particles (see Figure 1.13 in chapter 1).
- In calorimeters, a clustering algorithm is used to isolate the contributions of different particles to the detector response. Some clusters can be linked to the trajectory of a charged particle. Remaining clusters are associated to neutral particles, namely photons and neutrons. They are differentiated by their time of flight and the shape of the shower in the calorimeter.

[1] https://github.com/vkubarovsky/Model-for-the-pi0-eta-exclusive-cross-section

The reconstruction software [Ziegler et al., 2020] uses the CLARA framework for reconstruction and production of the final HIPO files, to be used for physics analyses.

3.1.2 Constitution of training datasets

Using the simulation software, we produce a dataset to feed machine learning algorithms for DVCS event selection.

3.1.2.1 Event preselection and choice of input variables

The physics analysis starts from the list of reconstructed particles, namely the list of identified particles with their four-momentum. Events that are kept must fulfill the following conditions:

- exactly one electron with an energy over 2 GeV (mandatory since it triggers the acquisition and to compute kinematic variables);
- at least one photon of energy over 0.8 GeV (to compute kinematic variables);
- the kinematics must comply with the phase space of interest being the deep inelastic regime, namely $Q^2 \geq 1.5\,\mathrm{GeV}^2$, $W^2 \geq 4\,\mathrm{GeV}^2$.

In practice, machine learning models that are used later in this thesis require a fixed input size. Therefore, the input vector is chosen to contain maximum five particles: the mandatory electron, an optional proton (it may have not been detected) and from one to three photons. The photons are ranked by increasing squared missing mass $ep \to ep\gamma$ (equals 0 for DVCS since no particle is missing and also 0 for π^0 production since the mass of the missing photon is 0). The three-momentum for all of these particles is included in the input vector in either coordinate system: Cartesian, spherical, or both. Therefore, missing values will be present in any sufficiently large data sample: either the proton remains undetected, or no background photon was emitted, leading to only one or two detected photons. A missing value does not necessarily mean that the particle has not been detected: it can simply mean that the particle never existed. Therefore, data imputation is not relevant.

3.1.2.2 Choice of phase space distribution: uniform or with cross-sections

We consider two strategies for data generation: taking into account the cross-sections models of DVCS and π^0 production, or generating data uniformly over the available CLAS12 phase space. Intuitively, a machine learning model trained on data with cross-sections risk to perform DVCS/π^0 production separation based on the relative dynamics of the two processes. Indeed, as shown on Figure 3.2, DVCS events are mostly localized in the low x_B region while π^0 production events are rather in the high x_B region.

Therefore, we develop an alternative generation strategy where both DVCS and π^0 production events are generated uniformly. We perform a uniform generation over the four-dimensional space (Q^2, x_B, t, ϕ), knowing that the upper and lower bounds of x_B depend on Q^2 and those of t depend on both Q^2 and x_B. However, ϕ is generated independently in [-180°, 180°].

We pass generated data, both with flat distribution and with cross-sections, into the simulation and reconstruction process. In the end, the nature of events retained after the preselection detailed above is presented in Table 3.1 for the flat generation and for

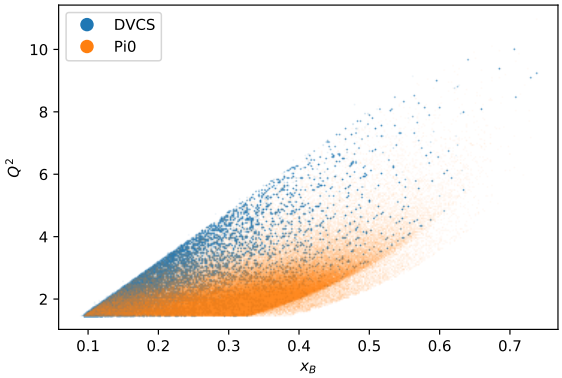


FIGURE 3.2: Localization of DVCS and π^0 production events in the (Q^2, x_B) phase space.

the generation with cross-sections. One thing to note specifically about π^0 production events is that the two photons are detected in about only half of the cases. In the other half, only one photon is detected, not to mention the numerous cases where no photon is detected, which are not represented in these tables.

TABLE 3.1: Percentages of events belonging to different categories. Top: flat generation; bottom: generation with cross-sections; for DVCS events on the left and π^0 production events on the right. The percentages are computed as function of the number of photons (noted γ in the table) and the presence or not of the proton.

Flat generation

DVCS: 84.8% of generated			
	No proton	Proton	
1γ	17.7%	44.8%	62.5%
2γ	6.1%	14.2%	20.3%
3γ	4.9%	12.3%	17.2%
	28.7%	71.3%	100%

π^0: 45.8% of generated			
	No proton	Proton	
1γ	11.8%	28.7%	40.5%
2γ	9.0%	23.5%	32.5%
3γ	7.5%	19.5%	27.0%
	28.3%	71.7%	100%

Generation with cross-sections

DVCS: 11.1% of generated			
	No proton	Proton	
1γ	43.0%	47.8%	90.8%
2γ	3.1%	4.5%	7.6%
3γ	0.7%	0.9%	1.6%
	46.8%	53.2%	100%

π^0: 16.2% of generated			
	No proton	Proton	
1γ	12.0%	39.8%	51.8%
2γ	10.8%	26.9%	37.7%
3γ	3.3%	7.2%	10.5%
	26.1%	73.9%	100%

Using the two generated datasets (with or without cross-sections), we train a FURIA rule base separately on each dataset with a size of 20000 instances (the justification for this dataset size is given in section 3.2). As input features, we provide the three-momenta of the input particles, plus a few high-level variables such as missing or

invariant masses. The FURIA base induced on the dataset generated without cross-sections (named "flat FURIA" in the following) comprises 16 rules, against 114 for the FURIA base induced on the dataset generated with cross-sections (named "cross-sections aware FURIA" in the following). Among the antecedents of all the rules for the flat FURIA, 45% use high-level variables, against only 23% for the cross-sections aware FURIA. Among the 55% other antecedents of the flat FURIA, 43% are still relevant variables for event selection, for instance the photons energies. These observations are summarized in Table 3.2.

Therefore, just observing the rule bases and the used variables confirms our intuition: the cross-sections aware FURIA base does not use discriminative variables that are mostly independent of the generation strategy, but rather the momenta of the particles that greatly vary depending on whether cross-sections were taken into account. For instance, the electron momentum and scattering angle θ are directly correlated with Q^2 and x_B. Moreover, many rules in the cross-sections aware FURIA base are dedicated to the split along the phase space, and very few to the physics discrimination. In other words, the model did not learn objective criteria that are independent of the generation.

TABLE 3.2: Accuracies of a FURIA rule base depending on the training and testing datasets.

	induced on flat	induced on cross-sections
Number of rules	**16**	114
Percentage of high-level variables	**45%**	23%
Accuracy on flat	**71%**	60%
Accuracy on cross-sections	82%	**94%**

Table 3.2 also gives the accuracies of the FURIA bases when applied on flat or cross-sections generated data. When applying the cross-sections aware FURIA to the flat generated data, the accuracy drops down to 60%. Nevertheless, the accuracy of the flat FURIA applied to cross-sections generated data is a 12% less than the score obtained by the cross-sections aware FURIA. Plotting the repartition of examples (with a flat generation) classified as signal and background as function of Q^2 and x_B, we obtain Figure 3.3. This is obvious that the cross-sections aware model based its selection on the cross-section distribution over the phase space. In the flat dataset used for testing, equal amounts of DVCS and π^0 production events are generated uniformly everywhere in the phase space.

From this last figure and the observation of the generated rules, we conclude that any machine learning model would be biased immediately if given the cross-section information. In regions dominated by a given process, the model is not able to isolate events of rare processes. In addition, rare events (for instance in high Q^2 regions) are very interesting from the physics point of view and should not be neglected in the training. It should be noted that a cut on the model's output, when possible, would permit to refine the selection purity in certain regions of the phase space. We conclude that a model is less biased if trained on a flat dataset than on a dataset generated with cross-sections. Therefore, all machine learning models in the remaining of this thesis will be trained on a dataset generated with a flat distribution over the phase space.

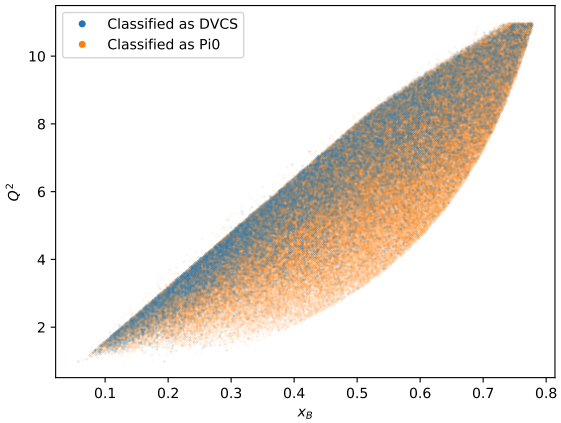


FIGURE 3.3: Phase space distribution of events classified by the FURIA model induced on data with cross-sections and applied on data generated without cross-sections.

3.2 Baselines using transparent models

Now that the training dataset is formed, baselines can be established using some of the transparent models presented in 2.2. First, we list the selected models.

3.2.1 Selected transparent models

C4.5

We use C4.5 decision tree as a first model that we implemented in Python. Missing values are handled as follows: an instance that has a missing value for the splitted attribute is spread into the sub-branches with a weight proportional to the number of data sent in each sub-branch. C4.5 usually obtains very good classification scores, at the expense of very large trees.

Fuzzy C4.5

We develop two fuzzified versions of C4.5 inspired from Olaru and Wehenkel [2003] and Chandra and Varghese [2009]. In both cases, we make the transition linear (see Figure 3.4:

- *Fuzzy C4.5 std* uses the principle of Chandra and Varghese [2009], namely setting the width of the fuzzy transition as function of the standard deviation of the attribute. In practice, we set $\beta = \alpha\sigma$ with α a hyperparameter of the algorithm.
- *Fuzzy C4.5 Fibo* determines β by Fibonacci search, following Olaru and Wehenkel [2003], to maximize the fuzzy information gain.

However, we noticed that this principle to split and then fuzzify raises an issue for ordinal attributes, i.e. attributes with a finite number of values but having an ordering, for instance the number of detected photons. Indeed, it can happen that the fuzzy split leading to the best information gain overlaps one of the values taken by the attribute. However, the same split will be repeated indefinitely in a very deep degenerate branch, so as to put apart a small proportion of instances taking the same value in a leaf at

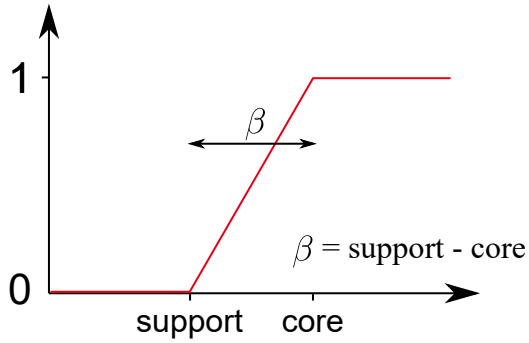


FIGURE 3.4: Illustration of the width β to determine for fuzzy splits in fuzzy C4.5.

each split. Such a degenerate branch is illustrated on Figure 3.5. The information gain of each split progressively becomes very small.

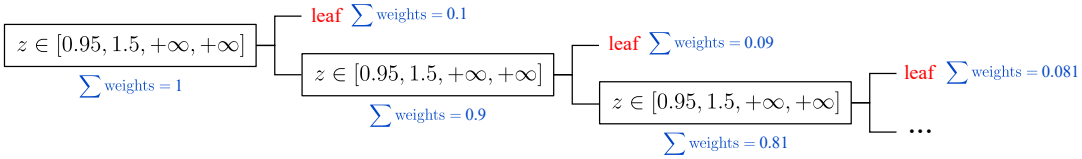


FIGURE 3.5: Degenerate fuzzy branch in case of an ordinal attribute z taking values $[0, 1, 2, 3, 4]$.

The problem arises from the fact that fuzzification is performed after the search of the split threshold. Indeed, fuzzification does not necessarily improves the information gain notably in the standard deviation version. This can lead to information gains superior to 0 but too small to be really significant regarding the number of instances or the number of classes. In addition, fuzzification does not have to respect the minimum number of instances per branch that is required when looking for the split threshold. That problem was neither mentioned by [Olaru and Wehenkel, 2003] nor by [Chandra and Varghese, 2009]. To solve it, we first set the minimal information gain to 10^{-5} to split the node instead of 0. We did not observe any split having an information gain below this value in our experiments, apart from the degenerate splits on ordinal attributes. In addition, we check the number of instances (as the sum of weights) in each child node also during fuzzification:

- for Fuzzy C4.5 std, we cancel fuzzification if it leads to children nodes without a sufficient number of instances;
- for Fuzzy C4.5 Fibo, we exclude from the Fibonacci search the widths that would violate the minimum instances per child node requirement.

Similarly to crisp C4.5, the fuzzy versions get good classification performances but often produce big trees that take advantage of big datasets, as displayed on Figure 3.6. However, fuzzy trees are often smaller, as displayed on the right plot of Figure 3.6.

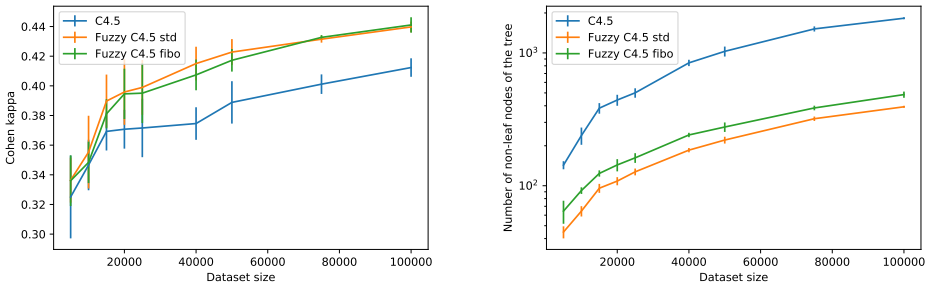


FIGURE 3.6: Left: Cohen's kappa score[2] of crisp and fuzzy versions of C4.5 as function of the dataset size. Right: the number of non-leaf nodes of the trees as function of the dataset size. The y axis is in log scale, since crisp trees are 3 times bigger than fuzzy trees for large datasets.

CART, AdaBoost and GradientBoosting

We utilize the CART, AdaBoost and GradientBoosting algorithms as they are implemented in the scikit-learn package [Pedregosa et al., 2011]. However, scikit-learn in general does not handle missing values and demands data imputation, which is not relevant in our case. In practice, using such models for CLAS12 would require to divide the data in separate training samples with different inputs depending on the present particles. However, we still consider these models as baselines and to compare notably with ensemble methods.

FURIA

FURIA is implemented in the Weka package in Java [Frank et al., 2016]. It has been reimplemented during this thesis in the ExpressIF® platform in C# developed at CEA[3]. In FURIA, instances with missing values will not be satisfying a rule involving the missing attribute(s). We noticed that FURIA does not necessarily take advantage of large datasets. Indeed, the FURIA algorithm is based on the rule covering principle, namely all instances of a given class must be covered by the rule base dedicated to this class, and no example of other classes must be covered by any rule of this rule base. This principle makes larger datasets more difficult to apprehend. This results in smaller rule bases and degraded classification performances when using too large datasets, as plotted on Figure 3.7.

Another weakness of FURIA is that it does not provide a continuous output between 0 and 1 contrary to a large number of machine learning models when dealing with binary classification. A given example is often covered by one or two rules of the same class and therefore assigned the output 0 or 1. Examples covered by at least one rule of each class are very rare. In the end, producing a ROC curve out of a FURIA model is impractical, making FURIA a model with poor flexibility when it comes to balancing between true positive rate and false positive rate.

[2]Cohen's kappa is a performance metric measuring the degree of agreement between two annotators (models in this case). It varies between -1 and 1: -1 means complete disagreement, 1 perfect agreement, and 0 is equivalent to random predictions.

[3]https://expressif.cea.fr/

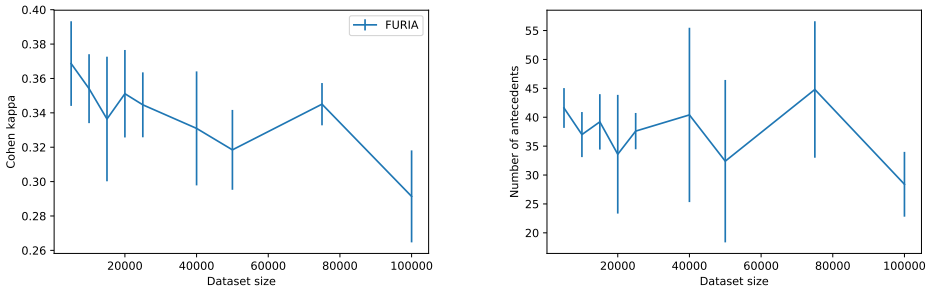


FIGURE 3.7: Left: Cohen's kappa score of FURIA as function of the dataset size. Right: the total number of antecedents in the base as function of the dataset size.

GAM

GAM are implemented in the InterpretML package in python using boosted decision trees as shape functions [Nori et al., 2019]. GAM with splines as shape functions are instead implemented in the mgcv package in R [Wood and Wood]. GAM do not natively handle missing values, however we propose the following solution: each term $f_j(x_j)$ will take a specific separate value for all instances having a missing value for x_j. Namely, a term f_j will be defined by a smooth univariate function (e.g. splines) over the range of x_j and by a constant value C_j that will be assigned to missing values of x_j. The C_j are determined over the training set to minimize the sum of squared errors.

3.2.2 Baselines

In the following, we use the Cohen's kappa[4] as performance metric. Indeed, we will deal with imbalanced datasets in part II and this metric permits a fair evaluation. We now obtain the first baselines training a few standard transparent algorithms on a 25000-events data sample. The results of a 5-fold cross-validation are presented in Table 3.3.

TABLE 3.3: Cohen's kappa obtained by a 5-fold cross-validation with a few transparent algorithms on CLAS12 simulated data with DVCS events as signal and π^0 production events as background. 25000 instances are used to obtain the results with missing values (first column) and 15000 instances without missing values (second column).

	With missing values	Without missing values
C4.5	0.369 ± 0.013	0.350 ± 0.015
Fuzzy C4.5 std	0.390 ± 0.018	**0.399 ± 0.018**
Fuzzy C4.5 Fibo	0.381 ± 0.010	0.395 ± 0.020
AdaBoost	//	0.285 ± 0.009
GradientBoosting	//	0.373 ± 0.015
FURIA	0.336 ± 0.036	0.281 ± 0.016
GAM with splines	**0.404 ± 0.017**	0.357 ± 0.014

[4]See footnote 2.

The best models on simulated data are the fuzzy C4.5 trees and the GAM. GradientBoosting also performs well on the dataset without missing values, but is obviously limited to it. C4.5 comes just after, followed by AdaBoost and FURIA. Regarding FURIA, it should be noted that the rule bases are much simpler than the trees generated by the crisp and fuzzy C4.5 trees, which is an asset for interpretability. Indeed, rule bases produced by FURIA are often incomplete, and rule stretching permits to classify uncovered examples. On the opposite, decision trees are by nature exhaustive. The evolution of the C4.5 score as function of the maximal depth is displayed on Figure 3.8. For the dataset with missing values, equalizing FURIA in terms of number of nodes or antecedents (37.6 ± 3.1 antecedents for FURIA at this dataset size) would set the maximal depth to $\lfloor \log_2(37.6) + 1 \rfloor = 6$. The Cohen's kappa obtained by such a tree on the dataset with missing values is 0.263 ± 0.028, largely smaller than the score obtained by FURIA.

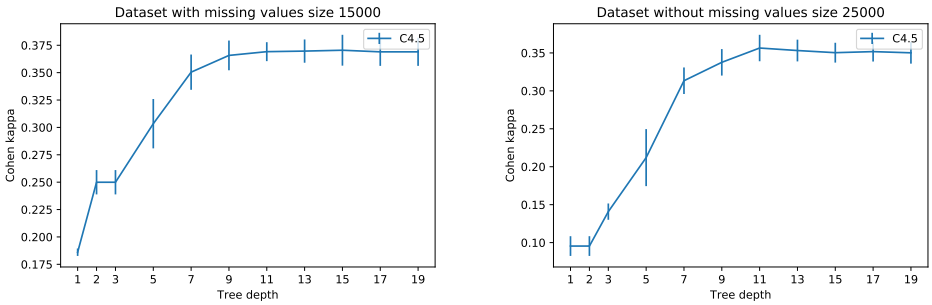


FIGURE 3.8: The evolution of Cohen's kappa score of C4.5 as function of its maximal allowed depth, for the dataset with missing values (left) and without missing values(right).

3.3 Proposed approach

The objective now is to improve the baselines scores of Table 3.3. The drawback of transparent models is often their lower classification performance compared to more complex models (Figure 2.5 in 2.1.4). Notably, they do not build complex internal representations of the training data. Therefore, next part (part II) focuses on the enhancement of the classification performance of different transparent models, through automated *feature construction* and inclusion of prior knowledge. However, the developed models are dedicated to be applied to real CLAS12 data analysis. Part III focuses on this objective: the models trained on simulated data are notably transposed to real CLAS12 data. The interpretability and selection performances of the models are assessed and compared to other approaches for the DVCS analysis.

Finally, we also attempted to exploit the knowledge of data imprecisions: indeed, the detectors are intrinsically imperfect and we have a limited knowledge of their performances. Errors arise both from the passage of particles through the detectors and from the reconstruction process. Fuzzy logic allows to consider fuzzy inputs: therefore, we adapted the crisp and fuzzy versions of C4.5 as well as FURIA to be capable of handling imprecise input. However, results obtained with these methods did not meet our expectations and we did not continue efforts in this direction. The details of our work on CLAS12 imprecisions handling in C4.5 and FURIA can be found in Appendix B.

Conclusion of part I

Chapter 1 introduced the physics context and challenges of the DVCS analysis conducted at CLAS12. Deeper information about the inner proton structure can be obtained thanks to correlation functions called generalized parton distributions, accessed through several exclusive inelastic processes including deeply virtual Compton scattering. This latter interaction is notably studied in the CLAS12 experiment that involves a series of detectors placed around a collision site between an accelerated electron beam and a fixed proton target. These detectors aim at covering the maximum available phase space while ensuring a reconstruction good enough to conduct the physics experimental program. The challenge is now to be able to distinguish properly the signal DVCS events from other background events, notably π^0 production events, to measure precisely the DVCS beam asymmetry. To deal with physical limits of the detectors and with limits of the standard physics analysis, artificial intelligence is proposed as a solution to optimize event selection, with a total transparency requirement.

Then, chapter 2 presented the field of interpretability in machine learning. Different types of interpretability coexist, notably transparency and post-hoc explainability. Classification, evaluation methods and limits of interpretability are discussed. Transparent models are then detailed, including decision trees, rule bases, generalized additive models with fuzzy variants. The remainder of this thesis will focus on these transparent models to perform DVCS event selection while being able to analyze the used model.

Finally, chapter 3 set the stage for the following parts by detailing the simulation process to constitute a training dataset, and by providing baseline scores using some transparent machine learning models. It was notably determined that machine learning models must be trained on datasets that are generated independently of the cross-sections to avoid biasing the training on the dynamics. This study was a first demonstration of the interest of using transparent machine learning models, since it provided additional arguments to dismiss one of the generation strategies.

The key points of this first part are:

1. Separating π^0 production events from DVCS events is the main challenge of DVCS event selection. Machine learning and especially transparent machine learning should help improving the selection by increasing the statistics and therefore reducing the statistical and systematic error due to background subtraction.

2. The transparency of certain machine learning models is a real asset to validate the robustness, evaluate errors, and inspect the learnt knowledge, but commonly comes with a degraded classification performance compared to models that are more opaque.

3. The training data directly impacts the knowledge that is learnt by a machine learning model. Notably, this also implies that simulation data must reflect as close as possible the reality. Otherwise, the selection performances will be degraded when analyzing real data. Most importantly, these performances will be poorly known, which is a major issue for the computation of cross-sections where the selection efficiency must be corrected for.

Each of these points will be discussed in the remaining of this thesis. The first point, i.e. the π^0 separation from DVCS, is the main goal of this entire thesis. The final analysis is performed in chapter 9. Part II is dedicated to tackling the second point to improve the classification performances of transparent models. Chapter 8 completes the discussion on the assets of interpretability specifically for CLAS12 data analysis. Finally, the third point on the importance of the training dataset and quality of the simulation has started to be studied in chapter 3 and will be further developed in chapter 7 where a domain adaptation from simulated data to real data is proposed.

Part II

Interpretable machine learning through feature construction

Introduction

Interpretable machine learning models often suffer from a decrease in performance compared to models that are more opaque [Došilović et al., 2018]. In this part, the objective is to increase the classification performance of transparent models. Their internal representation of data is often poorly elaborated, notably considering decision trees and rule bases that perform cuts of the unaltered input variables. In this part, we consider *feature construction*, a set of techniques aiming at building new features from the original ones.

No linear cut would be able to separate the two classes (orange and blue) in the example on Figure 3.9a. However, the computation of a new feature makes the problem solvable by a simple cut (see Figure 3.9b).

(A) y versus x.

(B) y versus $x^2 + y^2$.

FIGURE 3.9: Illustration of the computation of a new feature that greatly simplify the classification task for this toy dataset.

In this part, we propose a feature construction algorithm that is adapted to experimental physics datasets and apply it to a few classification tasks using several transparent models. Chapter 4 first reviews the state of the art in automated feature construction. Then, chapter 5 details the proposed feature construction algorithm adapted to experimental physics analyses that retains interpretability, and presents first experiments. Finally, chapter 6 deepens the experiments by embedding feature construction into decision trees, rule bases and GAM, the latter with additional adaptations.

Chapter 4

State of the art of feature construction

4.1 Tree-based feature construction algorithms 73
4.2 Evolutionary-based feature construction algorithms 74
4.2.1 Swarm intelligence 74
4.2.2 Genetic programming 75
4.3 Embedded feature construction 81

The choice of features can greatly affect the performances of machine learning algorithms, in particular the least complex [Sondhi, 2009]. In some cases, an inappropriate choice of features can even degrade the predictive power of a model [John et al., 1994].

The introduction of this part gave an example where adding a single feature permitted to solve perfectly a classification task. Note that adding new features is not the only way to improve the predictive capability: for instance, Support Vector Machines (SVM) [Vapnik, 1995] are able to linearly separate data after transforming the feature space through a kernel function.

The general field of *feature engineering* refers to any processing of the feature space, with the objective of ultimately using a feature set as input to a learning algorithm. Feature engineering gathers a large range of techniques, including feature extraction from raw data, data encoding, and feature transformation among others.

Feature selection and *feature construction* are two related subfields of feature engineering. While the objective of feature selection is to reduce the dimensionality of the feature set, notably to avoid overfitting, feature construction aims at building new features from the original ones, so that the performance of a learning algorithm is improved. Feature selection is widely studied in the literature (see [Chandrashekar and Sahin, 2014] for a survey). However, feature selection will never make up for an initial inappropriate feature set, while feature construction directly searches new relevant features for the learning task.

There exist a wide variety of feature construction algorithms, notably depending on the data type: for instance extracting useful features from pixelized images [Lim et al., 2007] or time series [Harvey and Todd, 2014]. Features can also be constructed indirectly by the learning algorithms themselves: SVM [Vapnik, 1995] use kernel functions to modify internally the feature space; the variables in the last layers of a deep neural network constitute good discriminating features as well [Bengio, 2013]. Principal Component Analysis (PCA) [Jolliffe, 2002] or Linear Discriminant Analysis (LDA) [Fisher,

1936] perform both dimensionality reduction and feature construction (i.e. combination of features).

However, features derived by these previous methods are often complex combinations of the primary ones and are not human readable, which is an issue in several sensitive applications (for instance health [Ambrosino et al., 1995]). One recent work [Kaul et al., 2017] uses correlations between feature pairs to add new features through a regression method. In the following, the focus is on explicit feature transformation, using a set of operators: Boolean (OR, AND, etc.), nominal (mean, number of occurrences, etc.) or numeric (+, −, log, etc.) among others. A constructed feature can thus be represented by a tree such as the one on Figure 4.1. In the context of physics applications, only numeric operators are used since the initial feature set consists in numeric values describing the state of the particles. However, a few studies using other types of operators will be included in this review. Not covered here but worth being mentioned, inductive logic programming is a class of feature construction methods returning Boolean features.

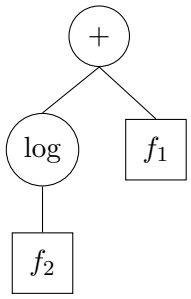


FIGURE 4.1: Tree representation of a feature, here $\log(f_2) + f_1$. The circles represent internal nodes and the squares represent leaves.

Depending on the classification algorithm and the candidate feature evaluation method, both feature construction and feature selection methods can be divided into three groups [Kohavi et al., 1997]:

- *Filter methods* compute a candidate feature score independently of the learning algorithm. For instance, information theoretic criteria such as information gain [Quinlan, 1986] or correlation with the output [Hall, 1999] may be computed to rank the candidate features and select the best one(s). Shafti and Pérez [2007] propose a comparison between several filter methods. It comes up with generic features that can be helpful for several learning algorithms at a time while avoiding the risk for overfitting. However, the final feature set may not be the most adapted regarding the chosen learning algorithm since the latter was not involved in the process.

- *Wrapper methods* use the prediction score of a learning algorithm to evaluate the candidate features. The main drawback of this class of methods is the computation time: it implies training and evaluating a model for each candidate, namely multiple times. This approach is therefore usually slower than a filter approach. However, the final feature set will be more specific to the learning algorithm, hence guaranteeing a better improvement of the performance compared to a set obtained through a filter method [Kohavi et al., 1997].

- *Embedded methods* combine the feature construction/selection process with the training of the machine learning model. These methods are usually fast but very specific to the chosen learning algorithm and hardly generalizable.

The different groups of algorithms for feature construction are presented in the next sections, as illustrated on Figure 4.2. A general survey can be found in [Sondhi, 2009] or in a more recent review focused on evolutionary methods [Swesi and Bakar, 2019]. Although filter and wrapper methods are presented together in their respective method-specific sections, a specific section (Section 4.3) is dedicated to embedded feature construction methods.

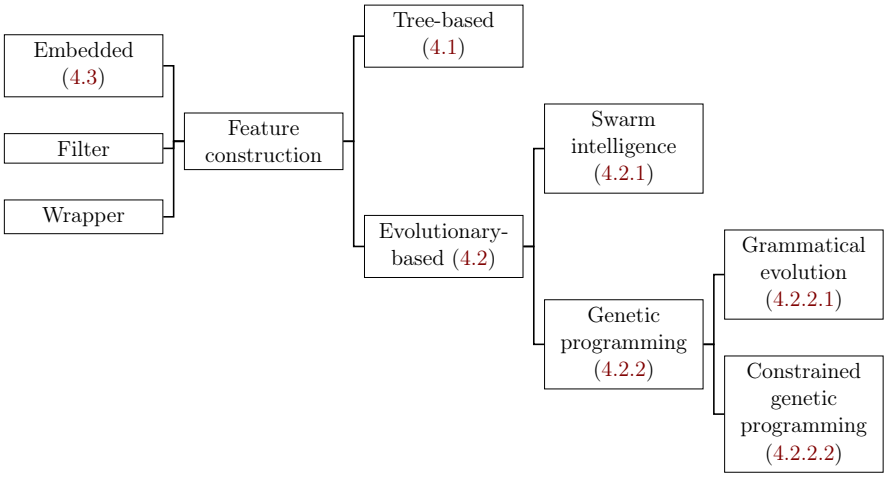


FIGURE 4.2: Classes of feature construction methods and structure of this chapter (in the parenthesis are the sections or subsections indicating where the topic is covered).

4.1 Tree-based feature construction algorithms

The first works on feature construction use mostly tree-based methods to search the feature space. The principle is to iteratively generate new features by combining those of the previous stage with the available operators (see Figure 4.3 for an example). The generation strategy (exhaustive or not, and with or without feature selection) varies between the different works.

The first tree-based algorithms such as FRINGE [Pagallo, 1989], CITRE [Matheus and Rendell, 1989] and DCFRINGE [Yang et al., 1991] use Boolean operators such as conjunction and disjunction. Then, FICUS [Markovitch and Rosenstein, 2002] generalized these previous methods notably by proposing numerical operators.

Several works use the strategy of expansion-reduction: new features are generated at each stage (for instance the exhaustive list of features that can be generated from the previous stage), and then a feature selection step is applied to keep only the most promising features for the next stage. TFC [Piramuthu and Sikora, 2009], FEADIS [Dor and Reich, 2012], DSM [Kanter and Veeramachaneni, 2015] and OneButton [Lam et al., 2017] apply this strategy. The main drawback is obviously the dimensionality of the feature set before selection that can become huge.

Maes et al. [2012] do not maintain a population of candidate features in their work, but instead grow iteratively one feature at a time using a Monte Carlo search embedded into the induction of a decision tree.

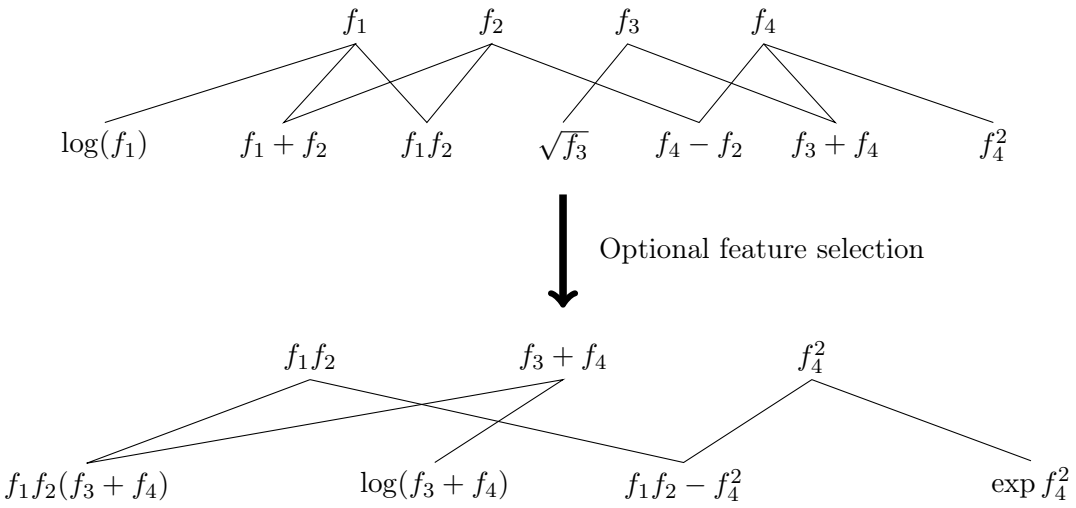


FIGURE 4.3: Example of a tree-based search for new features.

Khurana et al. [2016] propose Cognito, a framework to recommend univariate transformations to apply on a dataset, with a performance-guided exploration strategy of the transformation graph. Reinforcement learning is later applied to find the best exploration strategy [Khurana et al., 2018]. However, this method is memory-consuming and demands training on several different datasets.

More recently, the trend is to develop general feature engineering methods, i.e. that are applicable to any new dataset and learning task without specific retraining [Nargesian et al., 2017, Zhang et al., 2018].

While the most recent developments of tree-based algorithms overcome the drawbacks of the first works in the field, these methods are often memory-consuming. Moreover, since these methods are following a greedy approach, some good features may never be found if the path leading to them is cut in the early stages.

4.2 Evolutionary-based feature construction algorithms

Evolutionary algorithms are a class of metaheuristic algorithms inspired from biological evolution. The principle is to evolve a population of individuals that encode relevant information in their genes. Evolutionary computation applied to feature construction has been widely studied (see [Swesi and Bakar, 2019] for a survey). Two families of evolutionary computation methods are used for feature construction: swarm intelligence and genetic programming (GP), the latter being the most popular.

4.2.1 Swarm intelligence

A few works have been using particle swarm optimization (PSO) for feature construction. PSO [Kennedy and Eberhart, 1995] consists in a population of particles, each one being a candidate solution. Particles move in the search space to find the optimal solution. Each particle has a position X and a velocity V, each one being a vector of size D the dimension of the search space. Depending on the best individual position found by the particle P^i and on the best position found by its neighborhood P^n,

particles update their position and velocity:

$$V = \omega V + b_1(P^i - X) + b_2(P^n - X) \tag{4.1}$$
$$X = X + V \tag{4.2}$$

ω is the inertia parameter, b_1 and b_2 are randomly selected in the intervals $[0, c_1]$, $[0, c_2]$ respectively, with c_1 and c_2 as acceleration constants.

Xue et al. [2013] claim to be the first to propose a PSO based feature construction method. They use a binary version of the PSO algorithm to indicate the presence or absence of each base feature in the final constructed feature. Thus, each particle corresponds to one candidate feature, the dimensionality of the particle being the number of base features. This method does not include the choice of operators to combine the features. Dai et al. [2014] use the classical continuous version of PSO, and expand the dimensionality to include the selection of operators. They propose two representations: pair and array representation, to enable the evolution to optimize the operators as well as the features used:

- In the pair representation, the particles still have the dimensionality of the number of original features. Each element x of the position array of the particle corresponds to one feature: if $x < 0.5$ the feature is not used, otherwise the corresponding operator is chosen by dividing the interval $[0.5, 1]$ into equal parts.
- In the array representation, the size of the particle position vector is $2n - 1$, n being the number of initial features. Half of the vector is used to determine if the feature is used or not, and the other half to choose the operator associated to the feature in the case it is selected.

The advantage of the array representation over the pair is the independence of the choice of the operator with respect to the choice of using the feature. However, the pair representation is twice smaller and therefore simpler to optimize. Mahanipour and Nezamabadi-pour [2017] improve the previous algorithms by including a feature selection algorithm to reduce feature dimensionality before applying the PSO algorithm. Finally, Swesi [2020] adds feature clustering to reduce the dimensionality of the final feature set.

The main drawback of all of these PSO based methods is that the tree structure of the built features is highly constrained and has to be chosen before performing the PSO algorithm. The works presented above used only binary operators in a degenerate tree structure in which each node has at least one leaf child.

A recent method inspired by PSO exploits gravitational search algorithm (GSA) [Rashedi et al., 2009] for feature construction [Mahanipour and Nezamabadi-pour, 2019]. GSA is a swarm intelligence algorithm in which agents evolve and are subject to gravity. Each agent gets a mass corresponding to a heuristic evaluation. Agents attract each other by a gravity force, and progressively move towards the heavier agents (i.e. the good solutions). Multiple features can be built at a time, but this method still suffers the lack of flexible tree structure despite achieving good results.

4.2.2 Genetic programming

Genetic programming (GP) is an evolutionary computation technique comparable to genetic algorithms. However, while individuals in genetic algorithms are vectors,

individuals in GP are represented as trees [Koza, 1992]. Several representations that are more specific exist:

- linear GP [Banzhaf et al., 1998] with only a sequence of unary functions and one terminal, which is of little interest for feature construction purposes since correlations between features would not be exploited;
- Cartesian GP [Miller and Harding, 2008] in which individuals are represented as graphs, thus generalizing tree-based GP;
- grammar-based GP [Mckay et al., 2010a], which is more specifically covered in 4.2.2.2.

Tree-based GP evolves a population of individuals through specific mutation and crossover operations. The probabilities that crossover and/or mutation occur for a given individual are hyperparameters of the algorithm. Individuals are evaluated with a fitness function, and the poorly-scoring are eliminated from the next generation. Figure 4.4 summarizes the evolution process: a generation n of individuals undergoes crossover (exchanging branches) and mutation (modifying a branch); the offspring is evaluated and selection is performed to obtain the next generation.

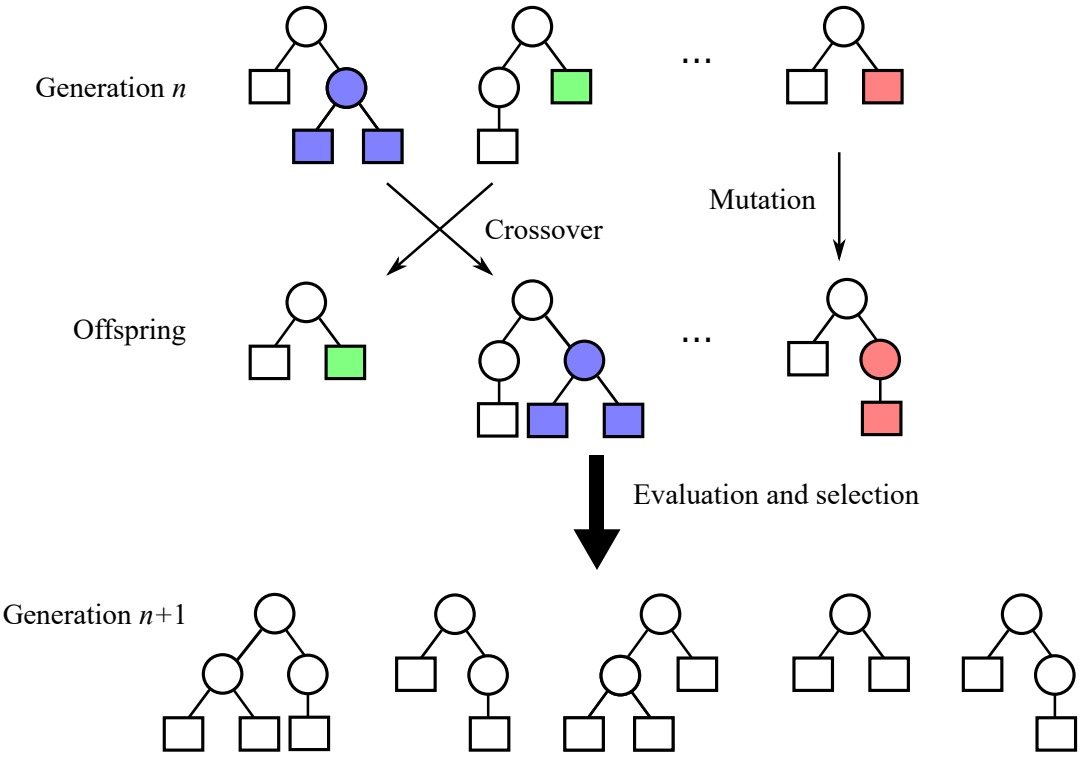


FIGURE 4.4: Overview of the principle of genetic programming.

The evolution operators (generation of the initial population, mutation, crossover, selection, and especially evaluation) can vary between the different studies, but the most used are:

- Generation, three different methods can actually be used:
  - *Full*: the tree is generated until a certain depth. All leaves reach this maximal depth.

  - *Grow*: for each node, unless it has reached the maximal depth, the node has a certain probability to be a leaf or else to begin a subtree.
  - *Half and half*: half of the time, the individual is generated using the "Full" algorithm, and the other half with the "Grow" algorithm.

- Mutation: the most common method is to regenerate a subtree, although many other methods exist (for instance replacing a node, generating a new branch or pruning one, etc.) [Koza, 1992].
- Crossover: two subtrees are exchanged between the two parents.
- Evaluation: depends on the application but involves a *fitness function*.
- Selection: usually a repeated tournament between k individuals: the individual with the best score among the k individuals is selected for the next generation. Compared to the strategy of selecting the best individuals, the tournament permits to introduce randomness into the selection process and to prevent quick convergence into a local optimum.

GP has been widely used as a feature construction method. The fitness function here serves to rank the individuals according to the fitness score of the corresponding candidate feature. The fitness function notably depends on the chosen evaluation approach: wrapper or filter. In addition, GP-based feature construction methods distinguish by the number of built features: some methods (mostly filter methods) add a single feature to the original feature set, while other methods produce multiple features and either add them to the original set or completely replace it.

Single feature construction

Various filter measures can be used for single feature construction. Otero et al. [2003] uses information gain ratio to evaluate the candidate features, while others prefer the Fisher criterion [Guo et al., 2005, Aslam et al., 2013, Ahmed et al., 2014b]. Muharram and Smith [2005] try four different filter measures for single feature construction. With the wrapper approach, Ahmed et al. [2014a] uses the classification score of a random forest to evaluate the candidate features.

Multiple feature construction

In most cases, single feature construction techniques add the newly built feature to the initial feature set. In contrast, build multiple features at a time is useful for classification problems for several reasons:

- adding more than one feature may add useful information to help the classification task;
- replacing entirely the initial set of features with a set of multiple built features can reduce the data dimensionality.

One approach is to perform several single feature construction processes: Neshatian et al. [2007, 2012] add sequentially one feature per class to discriminate this specific class from the others, using an entropy-based fitness function. Hart et al. [2017] also perform multiple feature constructions with the Hellinger distance (a distance between distributions) as filter fitness function, and then feature selection. This method is more computationally demanding since it requires more feature constructions than needed

and an additional feature selection step. Finally, Krawiec and Włodarski [2004] use cooperative coevolution to build multiple features: several single-tree populations evolve simultaneously. The fitness function of an individual in a given population is computed taking into account the best individuals in other populations, to maintain the complementarity between all the built features: each population is providing one built feature in the final feature set. This approach is more complex than the multi-tree representation techniques presented in the next paragraph.

Another method is thus to use a multi-tree representation for the individuals in GP: instead of having a single tree, an individual is now a fixed-length vector of trees and corresponds to a set of features. The evolution can then be done with a classical genetic algorithm [Vafaie and De Jong, 1995, Drozdz and Kwasnicka, 2010], with the mutation operator regenerating one of the trees, and for instance a single-point crossover operator. The other possibility is to use GP operators: the crossover exchanges two subtrees between two trees at the same location in the two parents, and the mutation operates only on one tree of the individual [Krawiec, 2002, Smith and Bull, 2005, 2007]. Tran et al. [2016b] build with this technique a number of features proportional to the number of classes, using a fitness function combining the accuracy of a classifier with a distance computed to minimize the intra-class distance and maximize the inter-class distance. The major drawback of these methods is that the number of built features becomes a hyperparameter of the algorithms.

The last method is to use a single-tree representation and to extract subtrees from it to form a multiple feature set. Guo et al. [2011] add a univariate function F to the list of operators for the GP algorithm. This function F denotes the start of a subtree that will be added as a feature in the final feature set. The advantage is that the number of built features is not fixed. Similarly, Ahmed et al. [2014b] extract all possible subtrees from the best individual to form a new feature set. Tran et al. [2016a] do the same but incorporating a feature selection step. However, they show later that a multi-tree representation generally gives better results than a single-tree representation for multiple feature construction, since interactions between features are better modeled in the multi-tree case [Tran et al., 2019].

In addition, a few works use Cartesian GP to perform multiple feature construction [Yazdani et al., 2017, Elola et al., 2017]. The graph representation of Cartesian GP allows to build a variable number of features at a time, but does not necessarily exploit the complementarity between features as in the multiple trees approach.

4.2.2.1 Grammatical evolution

Grammatical evolution (GE) is an evolutionary algorithm similar to GP, except that it uses grammars to generate the programs (or the features in the context of feature construction) [O'Neill and Ryan, 2001]. However, even if the generated programs take the form of trees, the evolution is different from the one of GP and is performed instead by a genetic algorithm. GE distinguishes indeed the genotype from the phenotype: the genotype is usually a list of integers that goes through the evolution process, and can be transformed into the phenotype, which is a tree expression used to get the fitness score.

Figure 4.5 illustrates the translation process from the genotype to the phenotype. The genotype is translated to the phenotype thanks to the context-free grammar associated with the problem. A context-free grammar does not depend on the context

and is defined by non-terminal symbols (that can be decomposed in a combination of symbols), terminal symbols (operators, variables or constants), and production rules (possible transformations from non-terminal symbols to other symbols). Figure 4.6 shows an example of a typical grammar used for feature construction that corresponds to the translation example on Figure 4.5. Starting from an individual (a list of integers, i.e. the genotype), the associated expression tree (i.e. the phenotype) is obtained by reading the individual left to right: an integer is translated into an expression by replacing the first non-terminal symbol by the n^{th} production rule of that non-terminal, n being the modulus of the integer by the number of rules.

Genotype: 08604221

start	<expr>
0 % 3 = 0	<expr> <op> <expr>
8 % 3 = 2	<term> <op> <expr>
6 % 2 = 0	<var> <op> <expr>
0 % n = 0	f1 <op> <expr>
4 % 4 = 0	f1 + <expr>
2 % 3 = 2	f1 + <term>
2 % 2 = 0	f1 + <var>
1 % n = 1	f1 + f2

Phenotype: $f_1 + f_2$

FIGURE 4.5: Illustration of the phenotype derivation from the grammar.

```
<start> ::= <expr>
<expr>  ::= <expr> <op> <expr> | <func> (<expr>) | <term>
<op>    ::= + | − | × | ÷
<func>  ::= cos | sin | tan | exp | log
<term>  ::= <var> | <const>
<var>   ::= f1 | f2 | ... | fn
<const> ::= 0 | 1 | ... | 9
```

FIGURE 4.6: Example of a context-free grammar used for feature construction with GE algorithms.

A few works use GE for feature construction. Gavrilis and Tsoulos [2005] say they are the first to propose GE to build a fix number of features with a wrapper method and continue their work in [Gavrilis et al., 2008]. Tzallas et al. [2016] apply this technique to electroencephalogram signals classification, using a radial basis function network as classifier to evaluate the individuals. Miquilini et al. [2016] extend this approach to single feature construction, hence with individuals of shorter length and easier to evolve. Gunawan et al. [2012] associate a probability to each production rule for a given non-terminal symbol, thus biasing the genotype generation.

However, Whigham et al. [2015] compare GE with grammar-based GP (covered in 4.2.2.2), and conclude that grammar-based GP outperforms GE in most of the problems. They also report that GE is often similar to a random search method. GE has indeed some drawbacks, including high redundancy (several individuals can lead to the same expression tree).

4.2.2.2 Constrained genetic programming

The next paragraphs are dedicated to the review of methods that constrain GP to either incorporate knowledge or improve interpretability. In many fields, there are requirements concerning the interpretability of the analysis method and/or the use of prior knowledge, which can guide the feature search [Doshi-Velez and Kim, 2017, Ratle and Sebag, 2001b]. Genetic programming offers many possibilities to enforce interpretability and/or to incorporate prior knowledge.

For instance, a few articles focus on improving the interpretability of the features *a posteriori*. Smith and Bull [2007] improve the readability of features with a parsimony measure, limiting their complexity. Several works use a multi-objective fitness function including a measure of simplicity (feature depth for instance) in addition to a performance metric [Icke and Rosenberg, 2011, Zhang and Rockett, 2011, Ahmed et al., 2016]. Similarly, Cano et al. [2017] facilitate data visualization as an additional objective.

Not limited to feature construction, several works constrain GP in its structure to incorporate prior knowledge. In his early work, Koza [1992] uses syntactic constraints to enforce the production of valid individuals. He notably restricts the crossover operation according to the operator types (Boolean, nominal, ...). Keijzer and Babovic [1999] deal with variables that have a dimension (there can be matrices or vectors for instance). They perform a multi-objective evolution to favor dimensionally consistent solutions, but non-valid individuals are still present in the population. Schmidt and Lipson [2009] use symbolic regression, containing a GP algorithm, to extract physics laws from experimental data. They do not consider the dimensionality of the input variables, letting constant scalar variables compensate for the units. They perform a multi-objective optimization with a parsimony measure.

In strongly-typed GP [Montana, 1995, Haynes et al., 1996], the crossover and mutation operators are restricted according to the constraints put on the construction process. Montana [1995] defines types to be assigned to each original feature. Every operator gets a list of accepted types (e.g. vectors, matrices, angles, etc.) to combine and a return type. Standard GP actually requires closure, namely that any individual tree can be considered as a subtree of another tree. To keep individuals valid during the entire evolution when constraining the construction process, the evolution functions are modified accordingly. For instance, the crossover operation is only possible between two subtrees of the same type. The mutation of a subtree should produce a new subtree of the same root type.

Constraints in GP can also be expressed through grammars: the technique is called grammar-based GP [Mckay et al., 2010b]. In grammar-based GP, contrary to GE, an individual is a tree derived from the grammar, called a derivation tree, from which one can infer a GP-regular expression tree by retrieving the leaves of the derivation tree. Examples of a derivation tree and an expression tree are displayed on Figure 4.7. The evolution uses the same operators than in standard GP, but applies on the derivation trees (the genotype) instead of the expression tree (the phenotype). However, the crossover and mutation operators are constrained so that the offspring respects the grammar.

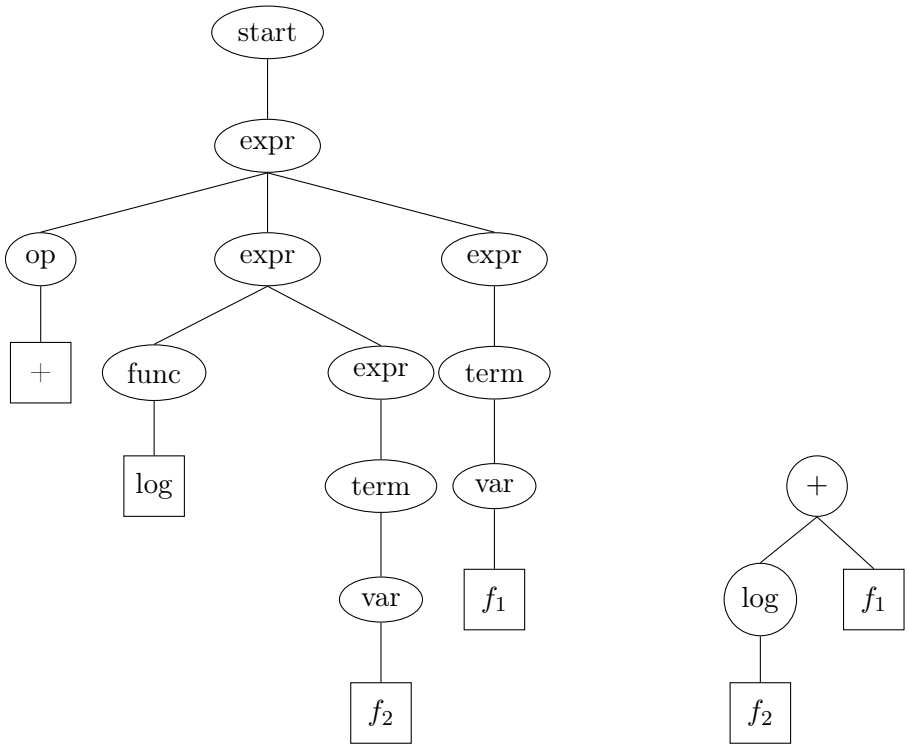


FIGURE 4.7: Derivation tree (on the left) and associated expression tree (on the right). The leaves of the derivation tree are read left to right to form the expression tree in a depth-first manner.

In [Ratle and Sebag, 2001b], a context-free grammar constrains the evolution to produce only individuals representing dimensionally valid expressions, to discover empirical physics laws from numerical experiments. They modify the initialization procedure, which can raise an error if there is not a terminal for each type. Crochepierre et al. [2020] use this principle for extraction of dimensionally-consistent features from sensor data.

In conclusion, grammar-based GP is a flexible method to enforce constraints and especially to consider the dimensionality of the input variables.

4.3 Embedded feature construction

The last section of this review is dedicated to embedded feature construction, in opposition to filter or wrapper methods, which are preprocessing methods. Indeed, prior feature construction may use a learning algorithm to compute the scores of the candidate features (if the wrapper approach is chosen), but the definitive model is trained only once the feature construction process is completed, using the final feature set. On the opposite, embedded feature construction permits to train the final model at the same time as the features are built. For instance, in a decision tree induction, each node would require its own feature construction process to get the splitting feature.

Few works are performing explicit numerical feature construction in an embedded manner. Most of the field is dedicated to performing feature construction in decision trees.

Ekárt and Márkus [2003] are the first to use a GP algorithm to find the best splitting feature at each node in the induction of a decision tree, with a custom error-based fitness function. In [Mugambi et al., 2004], a GP algorithm is also run in each node of a decision tree to find a polynomial of the initial features to be used as the splitting feature. The FCTree algorithm [Fan et al., 2010] randomly generates few candidate features in each step of the induction of a decision tree: the operators are weighted depending on their performance in previous iterations, and good operators have a higher probability to be picked later in the decision tree. In [Maes et al., 2012] a Monte Carlo search of features is embedded in several tree-based ensemble methods, building one feature at each node of the trees. The core idea is to explore a small part of the feature space at each decision node, given a set of constructor functions and the set of initial features. A simple search is performed to preserve the weak aspect of the base learner in ensemble methods. They suggest that the entire space of features may be explored in the whole training process by visiting only a small part of it at each node. Moreover, other optimization algorithms such as GP are avoided to prevent overfitting especially in the context of ensemble models. A randomized feature generation algorithm is performed at each node during tree growing. Zięba et al. [2016] also embed feature construction into tree ensembles (XGBoost in their cases): for a given tree, the considered features are combinations (with mathematical operators) of the most used features in the previous trees.

Finally, several works use genetic algorithms for both feature construction and model induction [García et al., 2014, 2015]. In a similar manner, Chen et al. [2017] use GP in a integrated manner to build the features and the model.

To conclude, embedded feature construction methods differ from the learning algorithm they are embedded into, but also from the complexity of the feature construction method itself: a number of methods avoid evolutionary algorithms by fear of overfitting [Maes et al., 2012].

In the two next chapters, we develop our own feature construction algorithm based on the state of the art, and apply it in a prior and embedded manner.

Chapter 5

Interpretable feature construction as a prior method

5.1 Constrained feature construction algorithm 84
5.1.1 Grammar-based GP to handle physical quantities 84
5.1.2 Transition matrix for guiding according to physics principles 85
5.1.3 Evolution 87
5.2 Experiments on prior feature construction 88
5.2.1 Experimental settings 88
5.2.2 Convergence of the algorithm 89
5.2.3 Performance comparison 90
5.2.4 Computation time 97
5.2.5 Discussion on interpretability 97
5.3 Conclusion and perspectives 100

As seen in the previous chapter, while most previous work in the feature construction field combine base variables without considering their potential units or dimensionalities, few works introduce grammars to control the validity of the built features [Ratle and Sebag, 2001b, Schmidt and Lipson, 2009].

This chapter introduces our feature construction algorithm adapted to physics data analysis by including prior knowledge about the built features: their units and dimensions are taken into account and a prior probability distribution is assumed in between the different mathematical operators. This chapter presents a generic method of constrained feature construction that can be applied to similar high-energy physics (HEP) event selection tasks and easily adapted to any problem that has similar prior knowledge, namely quantities with units and/or dimensions and prior knowledge about their relevant combinations. To demonstrate the performance of our method applied to other problems than CLAS12 event selection, all experiments presented in this chapter and the next one include up to three additional HEP datasets whose description is provided in Appendix C.

Section 5.1 details the general algorithm for constrained feature construction. This algorithm is used as a prior feature construction algorithm in the experiments conducted in section 5.2.

The method presented in section 5.1 and the experimental results for prior feature construction in section 5.2 have been published in the proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC):

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

5.1 Constrained feature construction algorithm

As seen in chapter 4 and particularly in 4.2.2.2, GP is the most flexible feature construction method to incorporate domain knowledge and notably for enforcing feature interpretability. Therefore, GP will be used and adapted for feature construction in the context of HEP applications.

HEP involves quantities that are each associated to a dimension and to a physical unit. For instance, input features are particle momenta in GeV/c, geometrical angles in degrees or radians, etc. When combining these base features into high-level variables through automated feature construction, some associations must be forbidden: for instance, adding an energy in GeV to a length in centimeters. Moreover, some recurrent patterns exist in the standard physics analyses that can be exploited to guide the search for relevant high-level features. In particular, we want to reproduce the approach of creating high-level features, common in fundamental sciences. Variables such as the missing masses or invariant masses are frequently used to select signal events, and consist in the square root of a sum of squared quantities (actually the norm of a four-momentum).

Therefore, two forms of prior knowledge can be used: first the knowledge of the base variables units and dimensions and second the experience of their usual combinations in physics analyses. Moreover, guiding the feature search as such should also permit to ensure interpretability of the finally built features: unit and dimension-consistent features are obviously mathematically valid and features that resemble in some ways to usual variables used by expert physicists are easier to read and more interpretable. These two forms of guidance are tackled respectively in subsections 5.1.1 and 5.1.2. The global algorithm is summarized in subsection 5.1.3.

5.1.1 Grammar-based GP to handle physical quantities

Grammar-based GP permits to enforce constraints during feature construction. Following the idea of Ratle and Sebag [2001b], we define a context-free grammar that constrains feature construction so that only valid features can be built form the point of view of units and dimensions. We limit the achievable unit powers so that the algorithm does not produce quantities with unit $\mathrm{GeV}^8/\mathrm{rad}^3$ for instance.

The grammar differs a bit depending on the dataset. For CLAS12, we make available the three-momentum as three independent components in Cartesian (p_x, p_y, p_z) and spherical coordinates (p_T, θ, ϕ). Finally, we also provide the 3D vector $\mathbf{p}$ as additional base variable. All base features, their dimensions and units are summarized in Table 5.1. For other datasets used in the experiments, only the features present in the dataset are used as base features. The grammar is defined accordingly. The descriptions of these datasets, their base features and grammars can be found in Appendix C.

CLAS12 grammar is presented on Figure 5.1. It includes standard operators such as addition, product, trigonometric functions and operations on vectors such as the

TABLE 5.1: Base features of each particle for CLAS12 data. Axis z is the incoming beam axis, p_T is the momentum in the transverse plane (x, y), θ the polar angle between the z axis and the transverse plane to the beam, ϕ the azimuthal angle between axes x and y. In total, there are 7 base features per particle for a total of 35 base features for CLAS12.

	Dimension	Unit
p_x	1	GeV/c
p_y	1	GeV/c
p_z	1	GeV/c
p_T	1	GeV/c
θ	1	rad
ϕ	1	rad
p	3	GeV/c

norm, scalar product, angle between two vectors, etc. A terminal can be either a base variable or a constant of the proper dimension/unit. We notably provided the proton mass $M_p = 0.938\,\mathrm{GeV/c^2}$ and the incoming electron beam energy $E = 10.6\,\mathrm{GeV}$ as constants. The grammar-based GP can be seen here as a strongly-typed GP, the types being the different unit and dimension configurations.

```
<start> ::= <E> | <E2> | <A> | <F>
<E>   ::= <E> + <E> | <E> − <E> | <E> × <F> | <E> ÷ <F>
   | sqrt(<E2>) | norm(<M>) | <component>(<M>) | <termE>
<E2> ::= <E2> + <E2> | <E2> − <E2> | <E2> × <F> | <E2> ÷ <F>
   | <E> × <E> | square(<E>) | dot(<M>,<M>) | <termE2>
<A>   ::= <A> + <A> | <A> − <A> | <A> × <F> | <A> ÷ <F>
   | angle(<M>,<M>) | <termA>
<F>   ::= <F> + <F> | <F> − <F> | <F> × <F> | <F> ÷ <F>
   | <E> ÷ <E> | <E2> ÷ <E2> | <A> ÷ <A> | sqrt(<F>) | square(<F>)
   | cos(<A>) | sin(<A>) | tan(<A>) | <termF>
<M>   ::= <M> + <M> | <M> − <M> | <termM>
<component> ::= get_x | get_y | get_z
```

FIGURE 5.1: Grammar used for CLAS12 data. E stands for a 1D momentum or energy in GeV/c or GeV, E2 for a squared momentum or energy in $\mathrm{GeV^2/c^2}$ or $\mathrm{GeV^2}$, A for an angle in radians, F for a unitless real number, M for a three-momentum of unit GeV/c. <termX> means a terminal of type X, namely a base feature or a constant.

5.1.2 Transition matrix for guiding according to physics principles

The default grammar-based GP algorithm randomly selects a production rule each time a non-terminal symbol must be transformed. For instance, the probabilities to transform the symbol <component> into get_x, get_y and get_z are equal. In HEP, operators are more frequent than others. For instance, the division is not often employed contrary to the addition. This is logical because physics analyses largely rely on energy and momentum conservation, i.e. summing the input and subtracting the output. The grammar as in Figure 5.1 already biases the search: trigonometric operators will be quite rare since they appear in only 3 out of the 13 production rules of <F>.

To enforce our own guidance on the feature search and favor the construction of formulas that are similar to those used in HEP, we choose a probability distribution on the transitions between operators. For instance, a square root in physics is very often followed by a sum of squares. In this way, we also forbid a square operator followed by a square root and conversely, to simplify the trees. This last constraint could actually appear in the grammar itself, but it would lead to a more complex and less readable grammar.

The transition matrix for CLAS12 is displayed on Table 5.2. These probabilities have been chosen manually following our expertise of the usual patterns in usual discriminative variables. The return type (i.e. the unit of the feature) is chosen among the one-dimensional types with a prior distribution. The first operator (i.e. the root of the tree) is chosen among all operators of the selected return type with a uniform distribution.

TABLE 5.2: Transition matrix for CLAS12 data. The probabilities are displayed for the next possible operations (as columns) given the previous operation (as row). Operations that are not listed as rows have a uniform transition probability distribution. Operations that are not listed as columns for a given previous operation cannot be selected as next operation (probability 0). The notations are the same than in the grammar (Figure 5.1).

Return type: {E: 0.5, A: 0.2, F: 0.1}.

	E + E	E - E	E × F	E ÷ F	sqrt(E2)
E + E	0.1	0.1	0.1	0.1	0.6
E - E	0.225	0.225	0.25	0.2	0.1

	E + E	E - E	E × F	E ÷ F	norm(M)
square(E)	0.5	0.1	0.07	0.03	0.3

	E2 + E2	E2 - E2	square(E)	E × E
E2 + E2	0.4	0.15	0.4	0.05
E2 - E2	0.2	0.07	0.7	0.03
sqrt(E2)	0.7	0.25	0	0.05

	E + E	E - E	E × F	E ÷ F	sqrt(E2)
E × F	0.15	0.15	0.35	0.3	0.05
E ÷ F	0.15	0.15	0.35	0.3	0.05

	F + F	F - F	F × F	square(F)	cos(A)	sin(A)	tan(A)
E × F	0.025	0.025	0.025	0.025	0.375	0.375	0.15
E ÷ F	0.025	0.025	0.025	0.025	0.1	0.1	0.7

	M + M	M - M
norm(M)	0.9	0.1

An attempt has been made to extract these probabilities automatically from physics documents: articles, books, lectures, etc. However, notations were often simplified and were mostly using vector notations instead of separate components. As an example, trigonometric functions barely appear. Therefore, this attempt to automate the design of the transition matrix failed and we kept our initial matrix (the one of Table 5.2).

Ratle and Sebag [2001a] set probabilities in the grammar (as opposed to our transition matrix between successive operators) and update them during the evolution. On the opposite, we choose not to update our transition matrix during the evolution. The idea behind introducing probabilities during the generation of individuals is indeed to guide the search towards physically sound features and not to seek higher performance.

5.1.3 Evolution

From the grammar and transition matrix, one can obtain valid tree-like individuals forming a population that can be evolved with GP. Since the grammar and the transition matrix are guiding the search, a few subtleties add to the standard GP evolution.

Contrary to GE, grammar-based GP directly evolves the trees. We consider both single and multiple feature construction: in the multiple feature construction case, we use a multi-tree representation and therefore evolve a population of individuals, which are lists of trees.

An initial population is firstly generated, evaluated and then evolved. For each individual in the population, mutation can be applied with a probability $P_{mutation}$ and crossover with a probability $P_{crossover}$. The offspring is then evaluated, and the selection is performed over the whole offspring and parent population, which has the advantage of keeping in the population some efficient features from the parent population.

The following paragraphs detail the evolution methods for both single and multiple feature construction.

5.1.3.1 Generation

Whether for constructing one or several trees (i.e. one or several features), the trees are generated using the half and half generation (see 4.2.2). The process of choosing the operators is however altered to respect the grammar.

A type T and the first operator (returning type T) are selected under the initial probability distribution. Then, while the condition on current depth is not satisfied, the tree keeps growing. The possible operators are selected according to the grammar. The transition matrix then defines a distribution probability among the possible operators according to the parent node. Finally, the leaves of the tree are randomly chosen among the set of base features and constants of the proper type.

After the population is initialized, the evolution process begins with a series of mutations, crossovers, evaluations and selections. However, the generation technique is still used during the evolution each time a tree or a subtree needs to be created, to keep following the same grammar rules and transition probabilities.

5.1.3.2 Mutation and crossover

Mutation and crossover operators apply classically at each generation of the GP algorithm to constitute the offspring.

The mutation method is randomly picked among three existing techniques. Each of these techniques is modified to be compatible with the grammar:

- *Uniform mutation*: a node is selected in the tree, then the subtree is entirely regenerated from that node while making sure that the dimensional consistency is still respected in particular at the root.
- *Node replacement*: a node is selected and replaced with any dimensionally compliant node.
- *Insertion*: from a selected node, a new subtree is inserted that have the original subtree(s) of the mutated node as child nodes. The inserted subtree is generated so that the grammar is also respected at the connections with the original tree.

The transition matrix is used each time a new tree or subtree needs to be generated to support the interpretability. In the case of multiple feature construction, mutation is applied on each tree of the list.

The crossover operation is the standard GP one-point crossover, assuming that the exchange of the two subtrees is compatible with the grammar, i.e. that the two roots of the subtrees share the same type. For multiple feature construction, the crossover is applied on the lists instead of on the trees: two sublists are exchanged.

The newly created individuals (either trees or lists of trees) form the offspring, which is to be evaluated.

5.1.3.3 Evaluation and selection

To evaluate an individual, trees are converted to numerical features by computing the function they represent on the base features. Any invalid operation (division by zero for instance) would create a missing value. Then, a fitness function must evaluate the transformed individual. To this end, several methods exist as seen in chapter 4. Wrapper methods evaluate the score (for instance accuracy) of a classifier trained with the newly built feature(s). Filter methods use ranking measures that are independent of any predictor.

The individuals of the offspring are therefore given a fitness. Selection applies on the joint set of individuals from the offspring and from the parent population. A repeated tournament selection among three randomly picked individuals constitutes the next generation population. The tournament size of three is a compromise between randomness and quick convergence.

5.2 Experiments on prior feature construction

This section presents experiments using the constrained feature construction algorithm detailed in section 5.1 as wrapper or filter feature construction method. For these experiments, three datasets are used: CLAS12 (without missing values), Higgs and $\tau \rightarrow 3\mu$. Their descriptions, grammars and transition matrices can be found in Appendix C.

5.2.1 Experimental settings

In the experiments, different versions of the proposed grammar-based GP algorithm are compared:

- standard GP (noted GP), without any constraint;
- grammar-based GP (noted GBGP), with the grammar;

- probabilistic grammar-based GP (noted PGBGP), with the grammar and the transition matrix.

The probability of mutation and crossover are both set to 0.6. Individuals can undergo both mutation and crossover during the same generation. The complexity of the built features is restrained by setting a maximal depth of 4 for the individuals (corresponding to maximum 15 nodes). An individual can consist of one or several trees, i.e. features. The experiments go from 1 to 6 built features. The GP evolution consists of a population of 500 individuals being evolved over 100 generations. The PSO method for feature construction proposed by Dai et al. [2014] and detailed in 4.2.1 is used as baseline for the proposed GP-based methods. Both the array and pair representations are tested. Similarly to the GP-based methods, 500 particles are evolved during 100 iterations. Other hyperparameters are the same as determined in [Dai et al., 2014]. Since PSO is only capable of building one feature at a time, it is compared only with the proposed methods for a single built feature.

In the following, several fitness functions are used. Filter and wrapper approaches are notably compared. For the wrapper approach, whatever the classification algorithm, its Cohen's kappa score over a 3-fold cross-validation performed on the joint set of base features and candidate feature(s) is used as fitness score.

The hyperparameters specific to each machine learning model can be found in Appendix D.

Evaluation is performed using a classification algorithm trained with the base features on the one hand and with the base features plus the newly built ones on the other hand. All results are presented with their mean and standard deviation over at least 25 independent runs (5 for each fold, for five folds).

5.2.2 Convergence of the algorithm

Figures 5.2, 5.3 and 5.4 display the evolution of the population fitness along the number of generations respectively for the CLAS12, Higgs and $\tau \rightarrow 3\mu$ datasets using CART as wrapper fitness function to build 1 or 6 features.

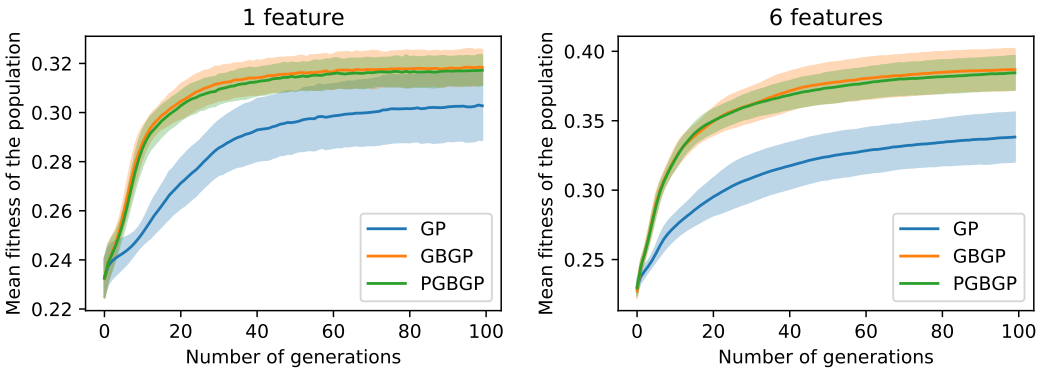


FIGURE 5.2: Convergence with CART as wrapper fitness function of the three variants of the GP algorithm for the CLAS12 dataset: mean fitness of the generation as function of the generation number.

In all cases, convergence is reached after 100 iterations or less. However, building 6 features require more generations than building a single one. Convergence is also

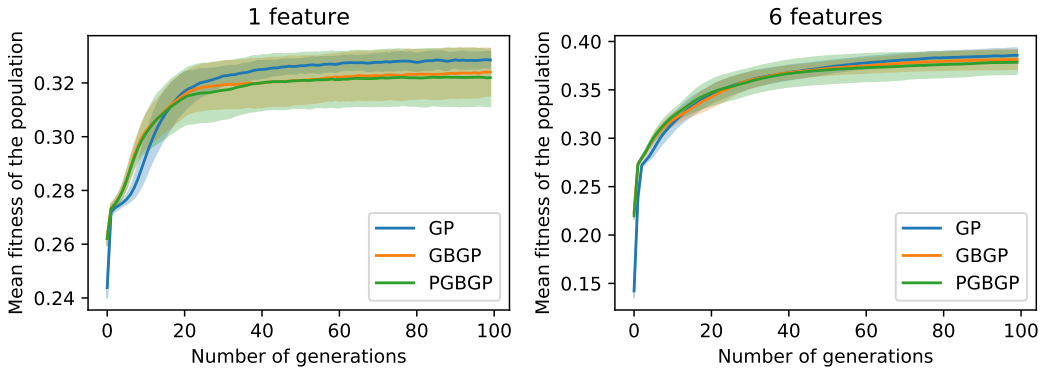


FIGURE 5.3: Convergence with CART as wrapper fitness function of the three variants of the GP algorithm for the Higgs dataset: mean fitness of the generation as function of the generation number.

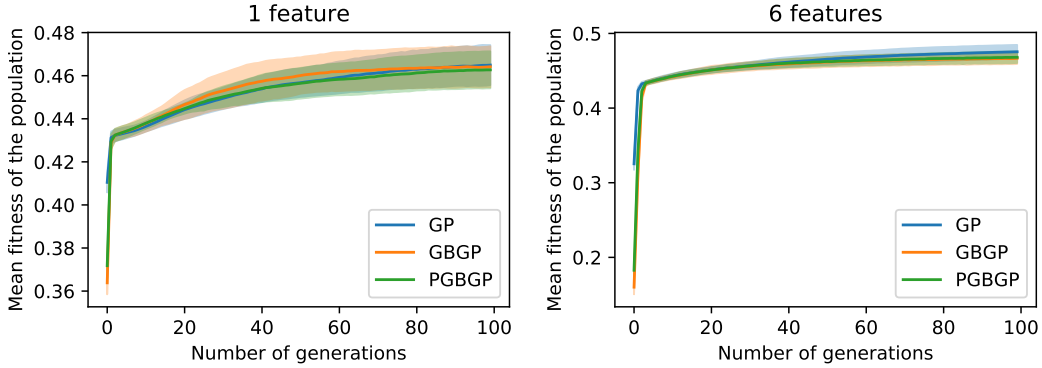


FIGURE 5.4: Convergence with CART as wrapper fitness function of the three variants of the GP algorithm for the $\tau \to 3\mu$ dataset: mean fitness of the generation as function of the generation number.

slower for CLAS12 than for other datasets, probably due to the larger number of base features plus the vectors. The convergence for the $\tau \to 3\mu$ dataset is fast in the very first generations. Then, the average fitness of the population hardly increases.

Therefore, setting the number of generations to 100 seems an appropriate choice for these datasets. To optimize the evolution, this number could be tuned depending on the dataset, evolution method and number of built features.

5.2.3 Performance comparison

In the experiments, three different fitness functions were tested: two wrapper approaches with CART and FURIA and one filter approach with the information gain. The main criterion for choosing the classifiers used for the wrapper fitness function is their rapidity of induction. This rapidity depends of course on the implementation, therefore C4.5 could be used as well as fitness function if the used Python implementation was as optimized as the CART implementation in scikit-learn. However, the induction of more sophisticated models like tree ensembles (AdaBoost, Gradient-Boosting) or GAM is intrinsically more complex and hence these models cannot be used as wrapper fitness function for feature construction in a reasonable amount of time.

5.2.3.1 Using the information gain as a filter fitness function

The information gain used in C4.5 decision tree induction is first used as a filter fitness function. Only one feature can be built with this technique since there is no control of inner correlations between features. Therefore, features built with this technique may be redundant in terms of discriminative information.

Table 5.3 compares the results obtained by the constrained GP versions (GBGP and PGBGP) with a few baselines for the three considered datasets. In all cases, feature construction permits to significantly improve the baselines: from 0.018 for $\tau \rightarrow$ to 0.070 for CLAS12. It is still difficult for the $\tau \rightarrow 3\mu$ dataset for which the PSO methods degrade the classification score. However, PSO pair performs particularly well for the CLAS12 dataset: actually, it is able to find most of the times a very good feature, namely:

$$p_z^e + p_z^p + p_z^{\gamma_1}. \tag{5.1}$$

This feature leads to an average gain of 0.078 on the Cohen's kappa score, which amounts to 20% of the baseline C4.5 score. It is very simple to build and therefore to find during a feature construction. It is also very often built by the constrained GP versions, GBGP and PGBGP. However, this result demonstrates a better stability of the PSO algorithm compared to the GP one for feature construction, since the unconstrained GP fails to find this feature most of the times, while PSO pair finds it in 97% of cases at least as part of a more complex feature.

Table 5.3: Cohen's kappa score for one built feature with different feature construction methods. Fitness function: information gain; evaluation: C4.5.

	CLAS12	Higgs	$\tau \rightarrow 3\mu$
Baseline (without feature construction)	0.318 ± 0.018	0.384 ± 0.006	0.470 ± 0.009
PSO pair	**0.383 ± 0.037**	0.351 ± 0.006	0.447 ± 0.036
PSO array	0.317 ± 0.108	0.343 ± 0.012	0.457 ± 0.019
GP	0.330 ± 0.034	**0.404 ± 0.013**	**0.488 ± 0.014**
GBGP	0.374 ± 0.026	0.395 ± 0.010	0.484 ± 0.010
PGBGP	**0.378 ± 0.020**	0.396 ± 0.013	0.483 ± 0.010

In other cases, the GP algorithms systematically obtain better results than the PSO based approaches. Then, the results are not significantly different between the unconstrained GP and the constrained versions except for CLAS12: GBGP and PGBGP surpass the simple GP method. This better outcome is probably due to the use of vectorized base variables for CLAS12. The unconstrained GP is incapable of handling them, while the vector data structure brings a significant contribution as prior knowledge to guide the feature search towards geometrical features.

Although the difference is not significant for other datasets, the simple GP algorithm outperforms slighly the constrained versions. Indeed, it has a larger search space available and should theoretically find equal or better features if given sufficient resources (for instance, population size and number of generations). However, the performance gap is not significant between unconstrained and constrained GP. This means that

the set of potentially interpretable features is as discriminative as the set of least understandable features built by the unconstrained GP.

Finally, the difference between GBGP and PGBGP is not significant, but these two methods are actually very similar in their conception, and only differ by the choice of the probabilities to construct the candidate features. It is therefore expected to not observe any significant variation in the distributions of the results. However, we hope to observe improvements regarding interpretability. This point will be discussed in 5.2.5.

Information gain is also a consistent fitness function to use to improve the classification performance of fuzzy C4.5, as displayed in Table 5.4. In this table are also displayed the classification scores of other classifiers, not necessarily linked to the information gain. For instance, CART, AdaBoost and GradientBoosting use the Gini impurity as discrimination measure and FURIA uses the FOIL's information gain. Using the built feature for other machine learning algorithms permits to increase significantly the classification score in all cases for CLAS12 and Higgs, except for AdaBoost with the Higgs dataset where the improvement is not statistically significant. However, building a feature with information gain as fitness function does not help other classifiers for the $\tau \to 3\mu$ dataset, except for CART and fuzzy C4.5 Fibo. These observations permit to conclude that the built feature with information gain has a good discriminative power generally and can be used in several contexts.

TABLE 5.4: Cohen's kappa score for one built feature with PGBGP and information gain as fitness function, using different classifiers for evaluation. "GB" stands for "GradientBoosting" in the table.

		CLAS12	Higgs	$\tau \to 3\mu$
Fuzzy C4.5 std	Baseline	0.345 ± 0.019	0.394 ± 0.004	**0.542 ± 0.004**
	PGBGP	**0.396 ± 0.020**	**0.410 ± 0.014**	0.453 ± 0.074
Fuzzy C4.5 Fibo	Baseline	0.356 ± 0.017	0.400 ± 0.003	0.545 ± 0.004
	PGBGP	**0.400 ± 0.019**	**0.416 ± 0.013**	**0.549 ± 0.008**
CART	Baseline	0.243 ± 0.016	0.276 ± 0.005	0.432 ± 0.007
	PGBGP	**0.294 ± 0.025**	**0.289 ± 0.010**	**0.443 ± 0.007**
AdaBoost	Baseline	0.333 ± 0.019	0.450 ± 0.014	0.643 ± 0.005
	PGBGP	**0.361 ± 0.017**	0.456 ± 0.010	0.645 ± 0.007
GB	Baseline	0.302 ± 0.012	0.387 ± 0.004	0.582 ± 0.006
	PGBGP	**0.396 ± 0.020**	**0.417 ± 0.012**	0.581 ± 0.024
FURIA	Baseline	0.236 ± 0.011	0.228 ± 0.021	**0.483 ± 0.011**
	PGBGP	**0.311 ± 0.030**	**0.306 ± 0.046**	0.464 ± 0.045
GAM	Baseline	0.320 ± 0.006	0.363 ± 0.007	**0.636 ± 0.005**
	PGBGP	**0.391 ± 0.014**	**0.395 ± 0.025**	0.631 ± 0.030

5.2.3.2 Using CART for the fitness function

The implementation of CART in scikit-learn is sufficiently fast to be used as a wrapper fitness function for feature construction. However, it does not handle missing values. Therefore, any feature leading to missing values (for instance, division by 0) will be discarded during the evolution.

Table 5.5 compares the scores obtained by the different feature construction methods while building one feature. Here, feature construction permits again to increase the classification score whatever the dataset or construction technique. In this case, GP outperforms PSO in all cases, including CLAS12: it seems that CART is a better fitness function than the information gain to guide the GP evolution. The constrained GP versions outperform the unconstrained GP for CLAS12 thanks again to the vectorized base features, with a gain of up to 0.020 for the PGBGP with respect to the unconstrained GP. The unconstrained GP beats slightly the constrained GP versions for Higgs and $\tau \to 3\mu$.

TABLE 5.5: Cohen's kappa score for one built feature with different feature construction methods. Fitness function: CART; evaluation: CART.

	CLAS12	Higgs	$\tau \to 3\mu$
Baseline (without feature construction)	0.243 ± 0.016	0.278 ± 0.007	0.432 ± 0.007
PSO pair	0.290 ± 0.020	0.307 ± 0.007	0.437 ± 0.008
PSO array	0.293 ± 0.021	0.306 ± 0.006	0.439 ± 0.008
GP	0.290 ± 0.020	**0.330 ± 0.006**	**0.462 ± 0.017**
GBGP	**0.310 ± 0.024**	**0.330 ± 0.010**	0.458 ± 0.012
PGBGP	0.300 ± 0.022	**0.327 ± 0.009**	0.455 ± 0.011

The main advantage of using CART as fitness function is that several features can be built at a time with GP. Since the built features must collectively improve the classification score of CART, this prevents them from being strongly correlated. Figure 5.5 displays the evolution of the Cohen's kappa score as function as the number of built features, for the three GP variants. PSO has here the drawback of not being able to build several features at a time. For CLAS12 and Higgs, the classification score increases with the number of built features. However, it does not for the $\tau \to 3\mu$ dataset: it seems that all discriminative information for improvement can already be contained in a single built feature. Similarly, the unconstrained GP for CLAS12 does not benefit from a larger number of built features. Maybe the gap is again simply due to the vector representation of base features exploited by the constrained GP, or maybe the unconstrained GP did not reached convergence: Figure 5.2 has indeed shown that the convergence of GP was not perfectly achieved when building 6 features. Therefore, the classification scores might be a bit better for GP. Regarding Higgs and $\tau \to 3\mu$, GP keeps performing slightly but not significantly better than the constrained versions while increasing the number of built features.

Features built using CART as fitness function are optimized to improve the classification score of CART models, but it does not mean that they are not generalizable to other machine learning models. First, AdaBoost and GradientBoosting use CART as individual learner. Therefore, their score improvement with feature construction should be comparable to the one of CART. Figure 5.6 displays the evolution of the Cohen's kappa score of AdaBoost and GradientBoosting trained using the features obtained by PGBGP with CART as fitness function. As expected, the global trends for the three datasets resemble those obtained with CART as evaluation model: the score raises until stagnating for CLAS12, keeps increasing for Higgs, and finally building many features does not seem to help the classification on the $\tau \to 3\mu$ dataset.

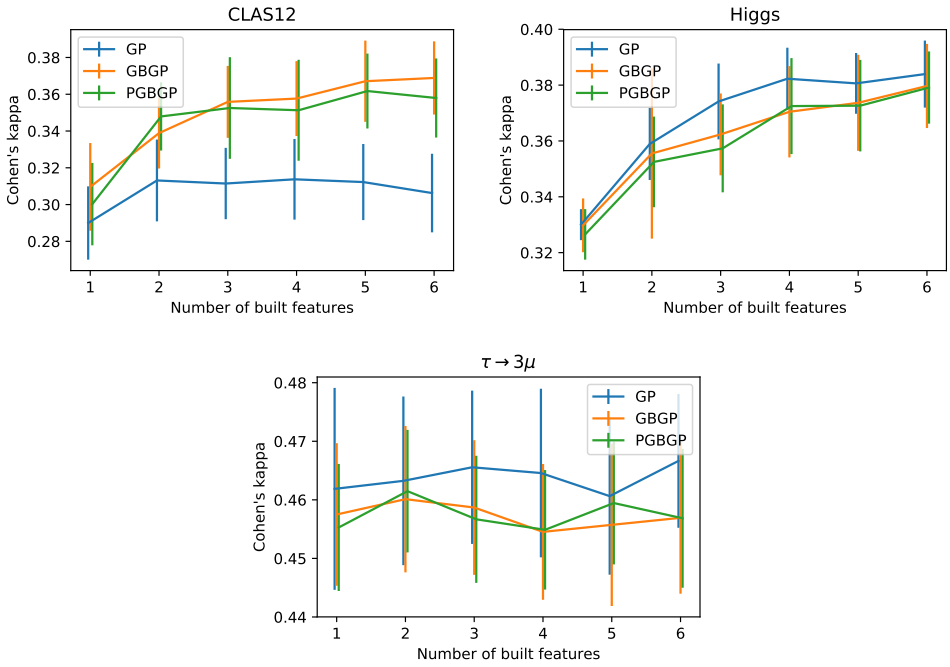


FIGURE 5.5: Evolution of the Cohen's kappa score with the number of built features with CART as fitness function and evaluation model.

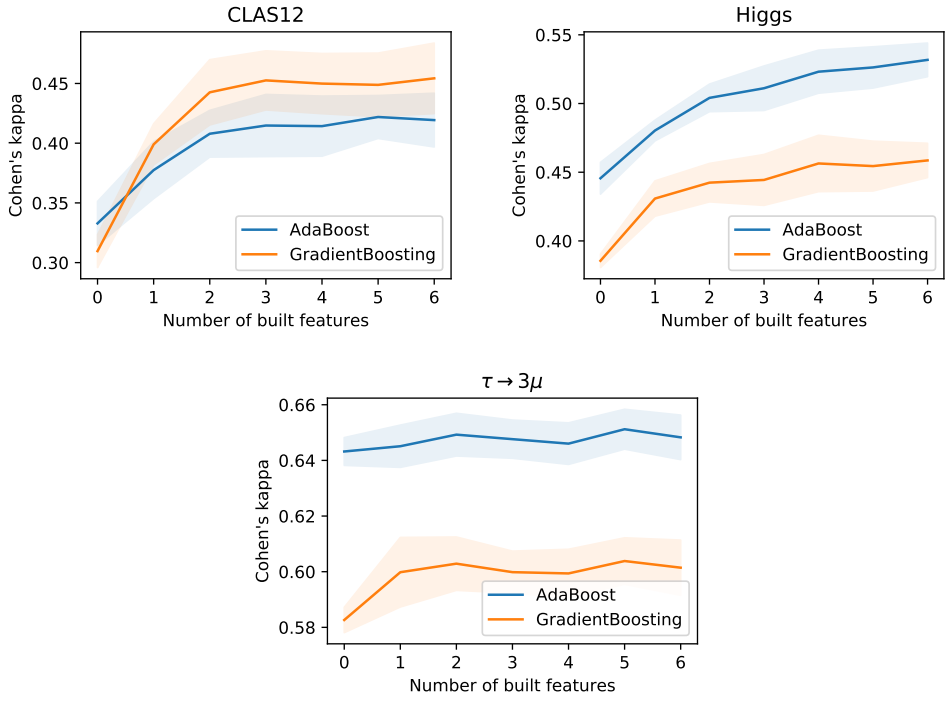


FIGURE 5.6: Evolution of the Cohen's kappa score with the number of built features by PGBGP with CART as fitness function and AdaBoost and GradientBoosting as evaluation models.

Figure 5.7 displays the scores using C4.5 and its fuzzy versions as evaluation methods. Using CART as fitness function seems to be beneficial as well to find features that are discriminant for C4.5, as least for the CLAS12 and Higgs datasets.

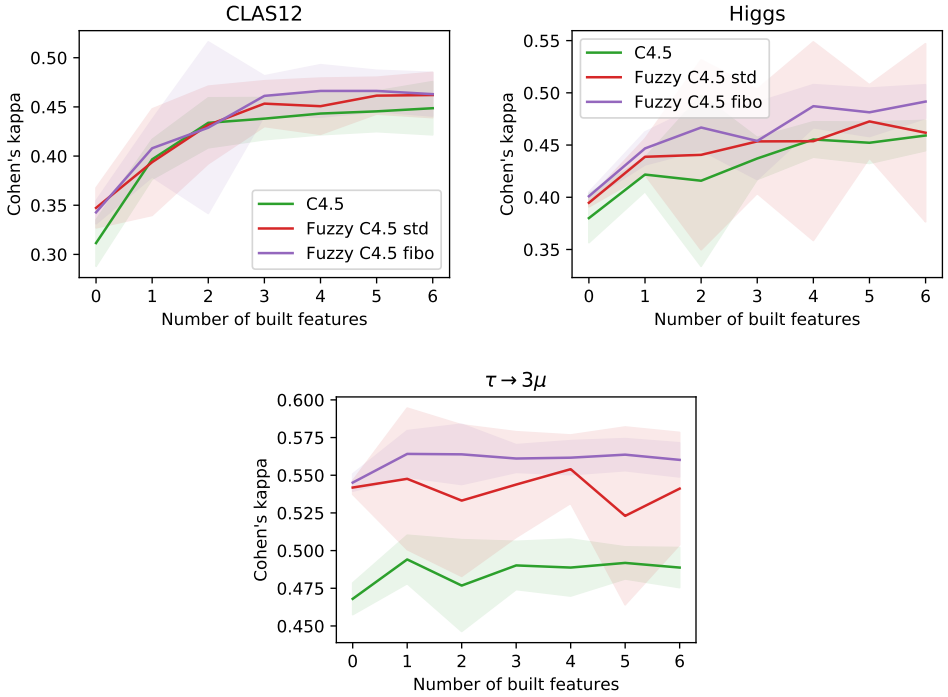


FIGURE 5.7: Evolution of the Cohen's kappa score with the number of built features by PGBGP with CART as fitness function and C4.5 (crisp and fuzzy) as evaluation model.

Finally, Figure 5.8 shows the results using FURIA and GAM as evaluation methods: these models are quite different from the decision trees used as fitness function, therefore they are relevant models to test the generalization power of the built features. For CLAS12, the built features improve significantly the classification score for GAM and FURIA, the latter in a more dramatic manner. The built features for Higgs also significantly improve the GAM classification performance, but not those of FURIA. Finally, the built features for the $\tau \to 3\mu$ dataset do not increase the classification scores of FURIA and GAM: these features are not generalizable.

5.2.3.3 Using FURIA for the fitness function

Finally, we perform experiments using FURIA as fitness function, for the CLAS12 dataset only. Table 5.6 presents the results. Again, we significantly increase the classification score with a gain of 0.127 (60% of the baseline) thanks to feature construction. The constrained GP versions outperform the unconstrained one. PSO feature construction methods are also largely behind GP based methods: the unconstrained GP improves by 28% the score of PSO pair.

Figure 5.9 displays the evolution of the Cohen's kappa score as function as the number of built features for FURIA. We observe again an improvement of the classification score with the number of built features but more moderately than previously. Indeed, the plateau is reached after only three built features for the constrained GP versions. Overall, we obtain a better score while using FURIA as fitness function than when using CART. For instance, building 5 features with CART as fitness function leads to a Cohen's kappa of 3.375 ± 0.037 with FURIA, while with FURIA as fitness function this score raises to 0.459 ± 0.010.

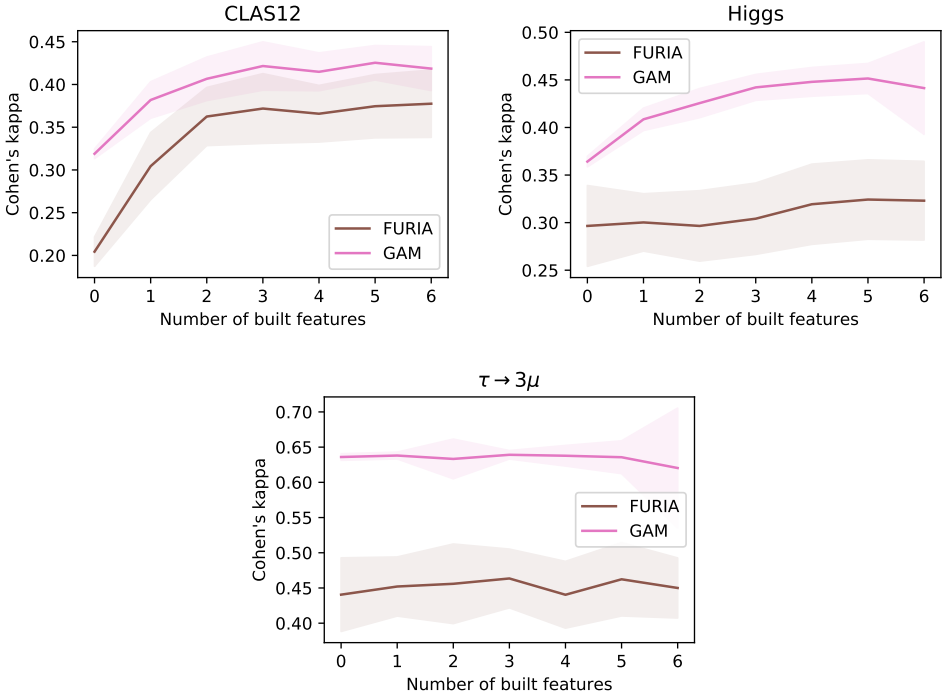


FIGURE 5.8: Evolution of the Cohen's kappa score with the number of built features by PGBGP with CART as fitness function and FURIA and GAM as evaluation models.

TABLE 5.6: Cohen's kappa score for one built feature with different feature construction methods. Fitness function: FURIA; evaluation: FURIA.

	CLAS12
Baseline (without feature construction)	0.212 ± 0.031
PSO pair	0.247 ± 0.039
PSO array	0.227 ± 0.044
GP	0.316 ± 0.061
GBGP	**0.337 ± 0.047**
PGBGP	**0.339 ± 0.041**

The constrained feature construction algorithm permits to globally increase the classification performances for all datasets, whatever the fitness function, except for the $\tau \to 3\mu$ dataset. Indeed, this dataset suffers from one major drawback: although the momentum and polar θ angle are available for the given particles, their azimuthal ϕ angles are not provided. Indeed, they are not discriminant since their distribution is often uniform over the range [-180°, 180°]. However, they are indispensable to build features that represent geometrical operations. Along with the $\tau \to 3\mu$ were already provided some handcrafted high-level features. With the features provided in the dataset, more complex non-linear and therefore non-interpretable combinations are probably more efficient to increase the classification score of a classifier. This can explain the better score of the simple GP, which is not constrained and then allowed to apply complex operations such as trigonometric functions on all variables. Therefore, this dataset is not very well suited for our feature construction algorithms.

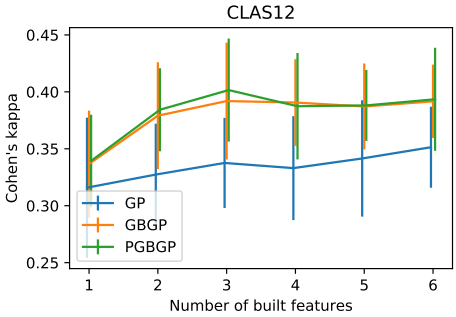


FIGURE 5.9: Evolution of the Cohen's kappa score with the number of built features with FURIA as fitness function and evaluation method.

5.2.4 Computation time

The computation time of the proposed feature construction techniques strongly depends on the kind of fitness function: while information gain is fast to evaluate, machine learning models such as CART or FURIA demand more time to be induced. Figure 5.10 displays the computation time taken per generation using PSO and GP for different numbers of features. The runs were performed with parallel computing on 8 to 12 cores. The computation time increases with the number of built features approximately linearly. PSO is twice faster than GP for entropy and CART and 20% more efficient for FURIA, probably due to the array representation instead of the more complex tree representation in GP. Finally, using entropy as fitness function is obviously faster than using CART. Although the results are presented here only for CLAS12, the trend remains the same for the two others. Our implementation of FURIA being less efficient than the implementation of CART in scikit-learn, we could not perform prior feature construction with FURIA as fitness function for the Higgs and $\tau \to 3\mu$ datasets.

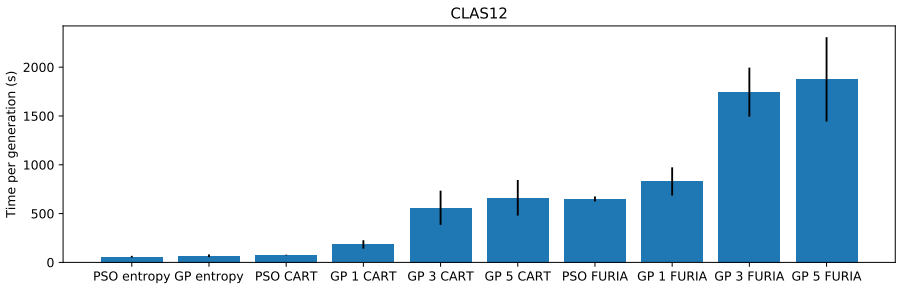


FIGURE 5.10: Computation time and standard deviation per generation for different feature construction methods and number of built features.

Figure 5.11 compares the computation time between the three variants of GP. Adding constraints to the GP algorithm, through grammars or transition matrices, increases complexity by a factor between 1.2 and 1.7 depending on the dataset.

5.2.5 Discussion on interpretability

In this work, we state that a feature is interpretable if a domain expert is able to recognize all or part of it and to understand why the feature is discriminative with respect

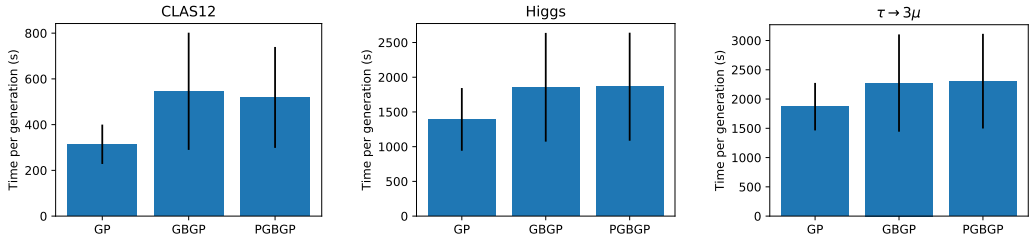


FIGURE 5.11: Computation time and standard deviation per generation for the three GP variants.

to the considered task. However, the feature is not necessarily known beforehand by the domain expert. This implies that the feature must be mathematically correct and that its unit can be inferred from its formula. In practice, interpretability is difficult to measure objectively, as discussed in chapter 2. However, we try to provide comparative elements in this part for the CLAS12 and Higgs datasets in particular. We do not focus on the interpretability of the $\tau \to 3\mu$ dataset, since the relevance of feature construction for this dataset is limited.

The following features have been built by the simple GP algorithm with CART as fitness function, the first two for CLAS12 and the next two for the Higgs dataset.

$$\cos(\sqrt{p_y^p} p_T^{\gamma_2}) + \frac{p_z^{\gamma_1} p_y^e}{\sin(\phi^{\gamma_2})} \tag{5.2}$$

$$\theta^{\gamma_2} p_z^{\gamma_1} \tag{5.3}$$

$$\sin(\cos(p_T^\tau)) \tag{5.4}$$

$$\cos(\phi^\tau - \phi^{lep}) \tag{5.5}$$

These features contain several inconsistencies from a physics interpretability point of view: series of trigonometric operations that should be applied only on angles, addition of a cosine with a product of momentum, etc. However, we notice for instance the $\cos\left(\phi^{lep} - \phi^\tau\right)$, which is perfectly valid and can actually be found among the features built by the constrained feature construction versions for the Higgs dataset. In addition, we mentioned that PSO pair for CLAS12 is building in most of the cases features that contain the following very discriminative pattern:

$$p_z^e + p_z^p + p_z^{\gamma_1}. \tag{5.6}$$

This feature is a quasi constant for DVCS (the beam energy), while it belongs to a continuous interval for background processes. Actually, this exact feature is built in only 10% of cases. In all other cases over the 97% where this pattern appears, it is combined with other operations, for instance:

$$p_z^e + p_z^p + p_z^{\gamma_1} + p_x^{\gamma_2}, \tag{5.7}$$

$$p_x^e - p_z^e + p_x^p - p_z^p - p_z^{\gamma_1} + p_x^{\gamma_2}.p_z^{\gamma_2} p_z^{\gamma_3} \tag{5.8}$$

Although the first of these two features respects the physical units, it is difficult to understand why adding $p_x^{\gamma_2}$ makes it more discriminative. The second feature is not valid from a physics viewpoint since it adds momentum (in GeV/c) with a product of

three momenta (in GeV^3/c^3).

However, these features could be used in practice for their discriminative power under the condition to fix the units of measurement for the input variables, since using other units could completely change the distribution of the features due to the different orders of magnitude of the variables.

The GBGP and PGBGP algorithms systematically build valid features, since they were designed for this purpose. Therefore, the features built by these algorithms are interpretable, but not necessarily as efficient as the non interpretable ones for the classification task. However, the interpretation process can still be complex.

The feature below has been built by the GBGP algorithm for the Higgs dataset:

$$\frac{\sqrt{p_t^{lep}\left(missingtE + p_t^{lep} + p_t^{\tau}\right)} + \sqrt{m_{H^0}^2 + missingtE^2 + p_t^{\tau 2}}}{\left(\cos\left(\phi^{lep} - \phi^{\tau}\right) + 2\right)^2 \cos^4\left(\theta^{lep} - \theta^{\tau}\right)} \tag{5.9}$$

The feature above respects the physical units, but does not have a direct intuitive meaning. However, although the ratio is not a usual operation in HEP, the elementary components can be interpreted. The numerator indeed resembles a transverse energy/momentum balance. In the denominator, the elements are more interpretable. As expected, the θ and ϕ angles are not mixed since they are defined in orthogonal planes. From energy/momentum conservation, a strong correlation exists between the direction of the lepton and the τ-lepton. Therefore, comparing the two angles of the two particles makes completely sense. Because the angles are defined within [-180°, 180°], the comparison is much easier with the cosine function.

For instance, the PGBGP builds the five following features for the Higgs dataset:

$$\cos\left(\phi^{lep} - \phi^{\tau}\right) \tag{5.10}$$

$$\cos\left(\theta^{lep} - \theta^{\tau}\right) \tag{5.11}$$

$$\cos\left(\phi^{missing} - \phi^{lep}\right) \tag{5.12}$$

$$p_T^{leading} \sum p_T^{jets} - \left(E_T^{missing} + p_T^{lep}\right)^2 \tag{5.13}$$

$$m_{H^0}^2 + \left(p_T^{lep} + p_T^{\tau}\right)^2 \tag{5.14}$$

The first and second features (5.10) (5.11) are exact elementary components of the single feature constructed by the GBGP. The third (5.12) is another geometrical combination of the angles of the missing transverse energy (most likely corresponding to a neutrino) and the lepton, which both come from the same τ lepton. It should be noted that the θ and ϕ angles are almost never mixed in the output features, proving that the GP algorithm inferred that these two angles belong to two different planes. The two last features (5.13) (5.14) are energy balances in the transverse space. These five constructed features are more interpretable features than the one from the GBGP, and are also independent of the system of measurement and therefore reusable in HEP analyses.

Regarding the CLAS12 dataset, the main asset of the GBGP and PGBGP over the simple GP for feature construction is that they have vectorized forms of the momenta

available, making geometrical operations much easier. Below are three features built together by the PGBGP algorithm for CLAS12:

$$\text{angle}(\overrightarrow{p^{\gamma_1}}, -\overrightarrow{p^e} + \overrightarrow{e^{in}} - \overrightarrow{p^{\gamma_1}} - \overrightarrow{p^p}) \tag{5.15}$$

$$p_z^p + p_z^{\gamma_1} + p_z^e + p_z^{\gamma_2} \tag{5.16}$$

$$\cos(\text{angle}(\overrightarrow{p^{\gamma_1}}, \overrightarrow{p^{\gamma_2}})) \tag{5.17}$$

with $\overrightarrow{e^{in}}$ the incoming beam momentum $(0, 0, 10.6)$. The second feature echoes the very discriminative one built by PSO, with $p_z^{\gamma_2}$ in addition. This still makes sense, since a π^0 production event will produce two photons, and such event is in most of the cases identified this way by physicists. The first and third features can actually be explained together. Indeed, the third feature observes the angle between the two photons if they exist: two photons by a π^0 decay are correlated, whereas if one of the photon is background this feature can take any value. The first feature compares the most energetic photon to the missing particle $ep \to ep\gamma_1$: if there exist a particle that has not been detected, this missing particle may be correlated with the first detected photon. In this case, the detected event is not a DVCS event since a DVCS event should only comprise the three particles e, p and γ_1.

5.3 Conclusion and perspectives

In this chapter, we have presented an adaptation of a new grammar-based GP algorithm to dimensional consistency and physics laws, aided with probability transition matrices that follow expert knowledge. Compared to a simple GP algorithm without constraints, we have shown that this method significantly improves the classification score of several classifiers for three high-energy physics datasets from completely different experimental setups. Nevertheless, the possibilities to build new features for the $\tau \to 3\mu$ dataset are limited due to the lack of base features. However, for the three datasets, our interpretable PGBGP-based feature construction algorithm brings a significant improvement on the classification score, up to 60% of the baseline for FURIA. We discussed the trade-off dilemma between performances and interpretability raised by the application of the transition matrix. The comparison between GBGP and PGBGP in gain in classification scores mainly relies on the data on which the experiment is conducted and on the number of features that are constructed. The interpretability for physics experts is definitely better with PGBGP, but the score may be better with GBGP (without probabilities) because of the need to compress information into a small number of features.

A perspective for this work would be to enforce the impact of probabilities as we go deeper in the generation of the tree: this could help building consistent elementary components while enabling more complex combinations at the root of the tree, imitating the high performing feature obtained by the GBGP on the Higgs dataset.

However, using the PGBGP as a prior construction method can be time-consuming. Therefore, we propose in the next chapter to use PGBGP as an embedded feature construction method.

Chapter 6

Interpretable embedded feature construction

6.1 Embedded feature construction in tree-based and sequential covering algorithms . . . 102
6.1.1 Principle . . . 102
6.1.2 Experiments . . . 104
6.1.3 Discussion on interpretability . . . 107
6.2 Boosting feature construction in generalized additive models . . . 111
6.2.1 Principle . . . 111
6.2.2 Bitonicity as prior knowledge . . . 112
6.2.3 Global GAM fitting method with embedded feature construction and bitonicity . . . 117
6.2.4 Experiments . . . 119
6.2.5 Discussion on interpretability . . . 124
6.3 Conclusion and perspectives . . . 127

Instead of using the constrained feature construction method presented in chapter 5 as a prior step before training a classifier, this chapter focuses on embedding the feature construction in the induction process of the classifier. This approach uses a filter fitness function, quicker to evaluate than a wrapper one. The constrained feature construction algorithm has been implemented in tree-based and rule-based algorithms (section 6.1) and in generalized additive models (section 6.2) with the goal to include prior knowledge.

The work on embedded feature construction in fuzzy decision trees detailed in section 6.1 has been accepted for publication in the proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC):

> N. Cherrier, J. Poli, M. Defurne and F. Sabatié, "Embedded Feature Construction in Fuzzy Decision Tree Induction for High Energy Physics Classification", *2020 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 615-622, IEEE, 2020.

The work on generalized additive models with embedded feature construction explained in section 6.2 has been accepted for publication in the proceedings of the 2020 International Conference on Discovery Science (DS):

N. Cherrier, M. Mayo, J. Poli, M. Defurne and F. Sabatié, "Interpretable Machine Learning with Bitonic Generalized Additive Models and Automatic Feature Construction", *International Conference on Discovery Science*, pages 386-402, Springer, Cham, 2020.

6.1 Embedded feature construction in tree-based and sequential covering algorithms

6.1.1 Principle

In this section, the fitness function is the split criterion in decision trees, tree ensemble methods or rule-based algorithms. One feature can potentially be built at each node of the tree(s) or at each condition of each rule in a rule base. However, Kotsiantis [2013] states that the decision complexity in decision trees decreases with the depth of the tree. In the context of oblique decision trees, linear models may generalize better, while non-linear models may lead to overfitting especially when used in deeper nodes [Yildiz and Alpaydin, 2001]. Therefore, just as ensemble methods restrict the complexity of individual classifiers, the number of feature constructions is limited and the construction algorithm is restrained. The method to embed feature construction is first presented for tree-based algorithms and then for sequential covering algorithms to produce rule bases.

6.1.1.1 In tree-based algorithms

The general algorithm for crisp or fuzzy decision tree induction with embedded feature construction is presented in Algorithm 2. A parameter N_{max} controls the maximum number of features that it is authorized to build. 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.

The *constructionCondition* can be for instance the condition in Equation (6.1) below: if it is satisfied for a depth d in the tree and a number of features built so far n_f, then the algorithm does not select an original attribute regarding the discrimination measure H but creates instead a new feature.

$$d \leq \log_2(1 + N_{max}) \quad \text{and} \quad n_f < N_{max}. \tag{6.1}$$

The 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.

Algorithm 2 is applicable to any tree-based algorithm following this scheme: for instance C4.5, CART, fuzzy decision trees, tree ensemble methods such as adaptive boosting or gradient boosting. For C4.5, the discrimination measure H is the information gain (possibly its fuzzy version for fuzzy C4.5), thus it is used as fitness function in the feature construction process. As for boosting algorithms, the fitness function varies. Boosting methods proceed iteratively by sequentially adding a new weak classifier to improve the overall performance. In adaptive boosting [Freund and Schapire, 1995] with decision trees, the weak classifiers are classification decision trees. What changes between the successive classifiers is the distribution of training examples: the latter are weighted depending on the performances of the previous classifiers. The Gini index used by the individual CART classifiers as discrimination measure is

Algorithm 2: Generic induction of a decision tree with embedded feature construction.

Input : *data*, subset of the dataset
n_f, 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
Output: a tree and the number of constructed features
Function `createTree`(*data, n_f, depth, N_{max}*)

```
r ← create a new node
if constructionCondition (n_f, depth) then
    A ← build new feature
    n_f ← n_f + 1
else
    A ← select the best original attribute regarding the discrimination
      measure H
Affect to r a test on A
M ← create m_1, ..., m_t a crisp or fuzzy partitioning for A
Split data into d_1, ..., 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 createTree (d_i, n_f, depth+1, N_max)
return (r, n_f)
```

then weighted. In the gradient tree boosting classifier [Friedman, 2001], regression trees are used as weak classifiers to fit residuals. The measure here is not binary but the mean squared error (MSE) between the residual and the mean of target values of the data in the considered node. H of a given feature and partitioning is the sum of the MSE of the children nodes that would be created by splitting on this feature.

The number of built features for a tree ensemble is already large if $N_{max} = 1$ for each individual tree. Instead, the parameter N_{max} is altered in this case to admit a probability to build a single feature at the root of the tree. It does not control in which tree a feature is built or not.

6.1.1.2 In sequential covering algorithms

Similarly to the embedding of feature construction in tree-based algorithms, a feature construction can be performed at each addition of an antecedent in a rule. As a reminder, sequential covering algorithms progressively add rules so that the training set is covered with a minimal set of rules.

In RIPPER and FURIA for instance, antecedents are added to a rule so as to maximize the FOIL's information gain (see 2.2.2). In the same way as for decision trees, a feature construction algorithm can be performed at this stage, the fitness function being the discrimination measure i.e. the FOIL's information gain.

The number of built features N_{max} can be limited as well. For a single rule, N_{max} varies from 0 to $+\infty$. The N_{max} features are built sequentially in the first antecedents of the rule. A rule base is treated the same way as a tree ensemble: N_{max} is the

number of built features per rule. To allow for a smaller number of built features in total, N_{max} can take values below 1 to reflect the probability to build a single feature in a given rule. There is no consideration of the order in which the rules are built.

6.1.2 Experiments

In these experiments on embedded feature construction in tree-based and rule-based models, we apply the different methods on the four datasets described in Appendix C: CLAS12 (without missing values), Higgs, $\tau \to 3\mu$ and MAGIC.

6.1.2.1 Baseline: Monte Carlo search

The work of Maes et al. [2012] is used as a baseline in the following experiments. Their algorithm is adapted to handle constraints like those encountered in HEP.

Maes et al. [2012] propose an embedded Monte Carlo search of features in tree-based models. The core idea is to explore a small part of the feature space at each decision node, given a set of constructor functions and the set of initial features, while preserving the weak aspect of the base learner in ensemble methods. They use a randomized feature generation algorithm at each node during tree growing.

A candidate feature is represented by its reverse polish notation (also called postfix notation). From the set of symbols containing the base features and the list of constructor functions, one can build an expression representing a valid candidate feature. The evaluation of one feature is done by numerically evaluating the expression and compute the discrimination measure used in the tree-based model at the current node.

Although Maes et al. [2012] present three Monte Carlo simulation strategies, the "step" variation is used in the experiments. Starting from an empty state, symbols are iteratively added. At a given iteration k, a random simulation, i.e. construction of a feature, is performed from the current state. The resulting feature is evaluated and its score compared to the best feature found so far. The symbol number k of the best feature found so far is added to the current state. The maximal length of a feature is a parameter of the algorithm but the simulation can be stopped earlier depending on the random simulation.

To constrain the Monte Carlo search, we adapt the computation of the set of valid actions depending on the current state. The set of allowed actions normally depends only on the length of the stack, the length of the current state and the arities of the operators. To constrain the search, the set of allowed actions must also depend on the units of measurement of the symbols. Each constructor function is assigned a set of valid input configurations, and each input feature is assigned a type (the same as in the constrained GP algorithm, e.g. energy, angle, unitless quantity). This assignment is directly linked to the grammar. With this modification, it may happen in simulations that no valid action exists for a given state. In this case, the current state is discarded and a new one starts.

We expect the Monte Carlo search to behave well while constructing one feature at each split node or rule antecedent, but we anticipate that the GP algorithm will already obtain good results when constructing features only at the first layers of the trees or first antecedents of the rules. A comparison of both feature construction techniques as well as a study of the impact of N_{max} are conducted in the following.

6.1.2.2 Experimental settings

In the experiments, we compare the proposed embedded constrained GP algorithm to the Monte Carlo search baseline. The same settings as in 5.2 are used for the constrained GP algorithm, although the fitness function changes.

For the Monte Carlo search, the same settings as in [Maes et al., 2012] are used, namely a budget of $100n$ evaluations for single decision trees and $10n$ for ensemble methods, with n the number of base features.

To be comparable with Monte Carlo search, a downgraded version of the constrained GP algorithm is proposed: only 6 generations are computed so that the number of evaluations of candidate features ($6 \times 500 = 3000$) is of the same order of magnitude for both Monte Carlo and GP methods.

The hyperparameters specific to each machine learning model can again be found in Appendix D.

The Cohen's kappa metric is used to compare the models. All results are presented with their mean and standard deviation over at least 25 independent runs (5 for each fold, for five folds).

6.1.2.3 Performance comparison

Many of the transparent models used before can be implemented with embedded feature construction following the principle detailed in 6.1.1. Although we present the results here only for C4.5 and FURIA, the complete experiments can be found in Appendix E: embedded feature construction scores have been computed for the fuzzy versions of C4.5 (std and Fibo), CART, AdaBoost and GradientBoosting.

C4.5

Table 6.1 compares the Cohen's kappa scores of different embedded feature construction methods in C4.5. The number of built features leading to the best Cohen's kappa score for the GP-based methods is specified in parentheses, while the Monte Carlo search builds 100 features. In all cases, performing embedded feature construction improves significantly the classification score, whatever the construction method. The downgraded GP algorithm performs better than the Monte Carlo search for all datasets except $\tau \to 3\mu$ for which both methods achieve a comparable score. Finally, the complete GP technique permits to improve significantly over the downgraded GP score. It should also be noticed that for two out of four datasets, the maximal score is obtained while building a limited number of features: from 15 to 20. This is a very interesting point concerning interpretability, since the complexity of the overall model is thus lower.

Figure 6.1 displays the evolution of the Cohen's kappa score for GP and downgraded GP while building from 0 (baseline) to 100 features. The score obtained by building features seems to reach a maximum, stagnating or even decreasing after a certain threshold of constructed features. Indeed, using 10 to 15 built features for CLAS12 already permits to reach a score that is not significantly far from the maximum. For Higgs and $\tau \to 3\mu$, the score seems to stagnate starting from 15 built features, but the score with 100 features is still higher. For MAGIC, the score with 100 built features is lower than with fewer than 25 features. In the end, it seems that the objective is reached: obtain a maximal classification score while building a limited number of

TABLE 6.1: Cohen's kappa score with different feature construction methods embedded into C4.5. The number of built features leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.316 ± 0.018	0.384 ± 0.005
MC	0.387 ± 0.018	0.419 ± 0.015
Downgraded GP	0.450 ± 0.024 (20)	0.442 ± 0.013 (100)
GP	**0.476 ± 0.025 (15)**	**0.461 ± 0.014 (100)**
	$\tau \rightarrow 3\mu$	MAGIC
Baseline (without feature construction)	0.470 ± 0.007	0.658 ± 0.010
MC	0.490 ± 0.008	0.674 ± 0.016
Downgraded GP	0.492 ± 0.010 (100)	0.693 ± 0.015 (20)
GP	**0.520 ± 0.010 (100)**	**0.705 ± 0.011 (20)**

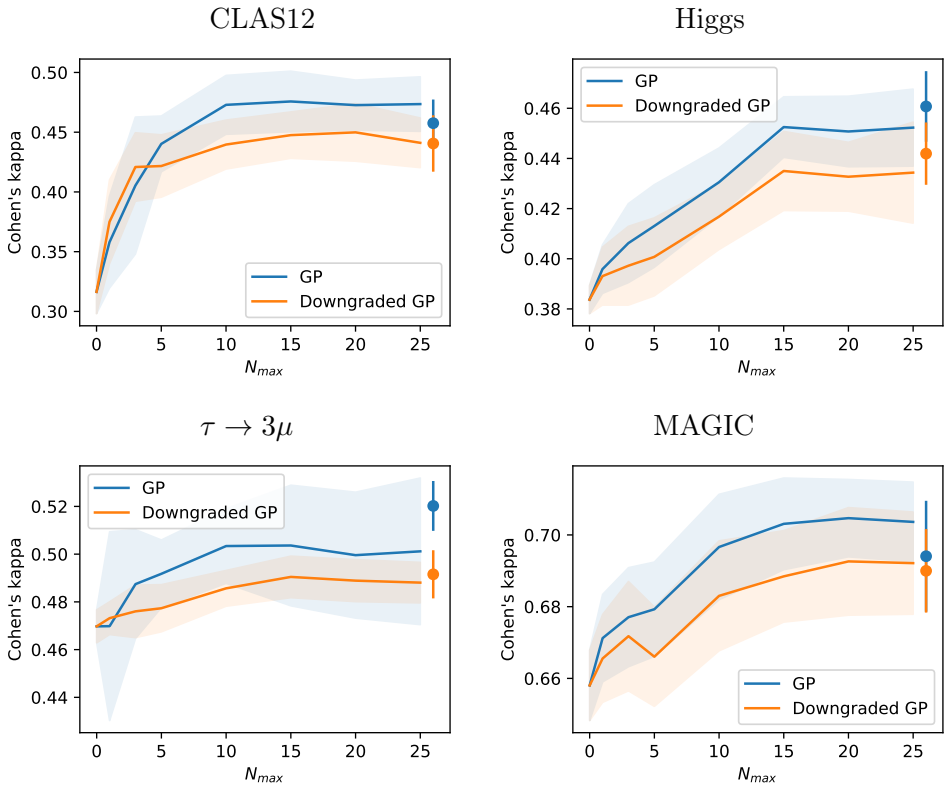


FIGURE 6.1: Evolution of the Cohen's kappa score with the number of built features in C4.5. The red marker on the right indicates the score obtained when building 100 features.

features. Indeed, while 100 features are impossible to apprehend together by a single person, it seems more feasible to analyze 10 or perhaps up to 15 features. A more thorough interpretability discussion is conducted in 6.1.3.

FURIA

We build features embedded in FURIA with the GP and downgraded GP versions. Table 6.2 details the results for the four usual datasets. For the GP algorithm, we go up to 3 features per rule for MAGIC, 2 for CLAS12, and one feature per rule for Higgs and $\tau \rightarrow 3\mu$ for computing time reasons. For the downgraded GP algorithm, up to 10 features per rule can be built.

TABLE 6.2: Cohen's kappa score with different feature construction methods embedded into FURIA. The number of built features per rule leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.236 ± 0.011	0.228 ± 0.021
Downgraded GP	0.490 ± 0.017 (2)	**0.387 ± 0.053 (4)**
GP	**0.496 ± 0.039 (2)**	0.350 ± 0.091 (1)
	$\tau \rightarrow 3\mu$	MAGIC
Baseline (without feature construction)	0.483 ± 0.011	0.498 ± 0.035
Downgraded GP	**0.490 ± 0.083 (2)**	**0.660 ± 0.022 (10)**
GP	**0.490 ± 0.075 (1)**	0.602 ± 0.042 (3)

Again, impressive improvements of the baseline classification scores are obtained thanks to the feature construction. The GP algorithm does not necessarily lead to better results compared to its downgraded counterpart. Figure 6.2 displays the evolution of the classification score as a function of the number of built features per rule. Depending on the dataset, a maximal score is reached at some point: for CLAS12 and Higgs notably, building more than 2 or 3 features is useless if not counterproductive. Building too many features may indeed lead to overfitting. On the contrary, the MAGIC dataset does not seem to suffer from this limit. However, restraining the number of built features is an asset for the interpretability of the model overall. Finally, only a small improvement is observed with one (GP) or two (downgraded GP) built features per rule for the $\tau \rightarrow 3\mu$ dataset.

Here, the optimization step of the FURIA induction algorithm adds complexity: instead of building rules once and for all, the optimization considers a replacement rule and a revision rule for each rule in the base [Hühn and Hüllermeier, 2009]. Therefore, the number of feature constructions to perform is multiplied. Figure 6.3 displays the number of rules and feature constructions as a function of N_{max} the maximum number of built features per rule. Because of this optimization step (done twice in our settings), the number of performed feature constructions per rule is approximately three times N_{max}, which makes the embedded feature construction impractical for large rule bases.

6.1.3 Discussion on interpretability

Model conciseness is supported by the smaller number of built features required by the models to get an optimal classification score. The final feature space obtained

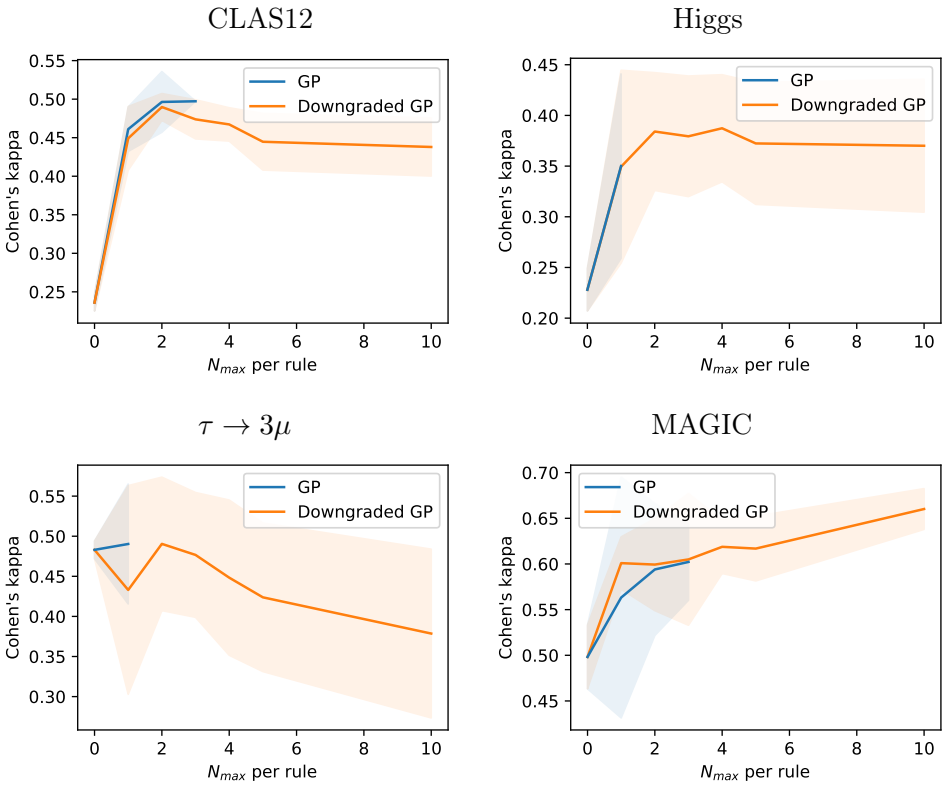


FIGURE 6.2: Evolution of the Cohen's kappa score with the number of built features per rule in FURIA.

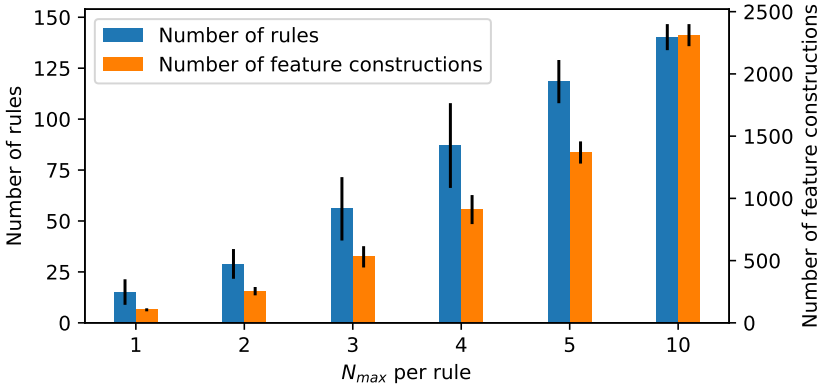


FIGURE 6.3: Number of rules and feature constructions with their standard deviations for the CLAS12 dataset as a function of the number of built features per rule.

with embedded feature construction is hence of reduced size compared to other feature construction methods. The features themselves are interpretable thanks to the constraints put on the GP algorithm, as discussed in 5.2.5. Moreover, they are globally discriminative for the classification problem especially if they are used in the first nodes of the trees, since they are relevant for the separation of bigger data subsets.

Equations (6.2) and (6.3) 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):

$$\cos\left(\theta^{lep} - \theta^{\tau}\right) \qquad (6.2) \qquad\qquad \cos\left(\phi^{lep} - \phi^{\tau}\right) \qquad (6.3)$$

These features visibly compare the geometrical angles of a lepton and a τ, two particles that are indeed expected to be the products of a single interaction: either a Z-boson disintegration into two τ leptons (the Z-boson may come from a Higgs boson), or the decay of two top quarks, or misidentified particles coming from the decay of a W boson. Comparing the geometrical positions of these two particles gives clues on their origin.

Equation (6.4) shows a recurrent feature built for the DVCS dataset (it appears in 79% of runs):

$$p_z^e + p_z^{\gamma_1} + p_z^p \qquad (6.4)$$

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.

However, these features were especially created in the first layers of the trees. They are globally discriminative for the considered problems. When looking more specifically at features built at deeper nodes of the trees, the interpretation is trickier. The following feature has been built in a fuzzy C4.5 Fibo for CLAS12 at depth 6:

$$-p_T^e + 2p_T^{\gamma_1} + p_z^{\gamma_2} + p_z^e + \|\overrightarrow{p^p}\|. \qquad (6.5)$$

While it respects the physical units, its physical meaning is more complex to understand. Two parts can be distinguished: one the one hand a sum of transverse momenta, and on the other hand a sum of z momenta. The norm of the proton momentum may contribute to both sums. This feature was built based on the samples that reached the node, namely 365 samples i.e. 2.4% of data. It is logical that this feature is more specialized to this selected sample and therefore less easily interpretable.

Regarding the model complexity itself, Figure 6.4 displays the number of nodes in the trees as function of the number of built features. It also includes fuzzy variants of C4.5. No correlation can be inferred between the size of the tree and the number of built features. However, building features permit to decrease the size of the tree for CLAS12 and MAGIC. In addition, the fuzzy versions of C4.5 are significantly smaller than its crisp counterpart except for $\tau \to 3\mu$. Of course, the size of the trees can be limited a posteriori by stopping the classification at a chosen maximal depth.

Regarding FURIA, the produced rule bases become more complex with the number of built features (see Figure 6.3), which is a major downside. Of course, we could still consider deleting the last antecedents of the rules, even if it is not intended by the FURIA algorithm.

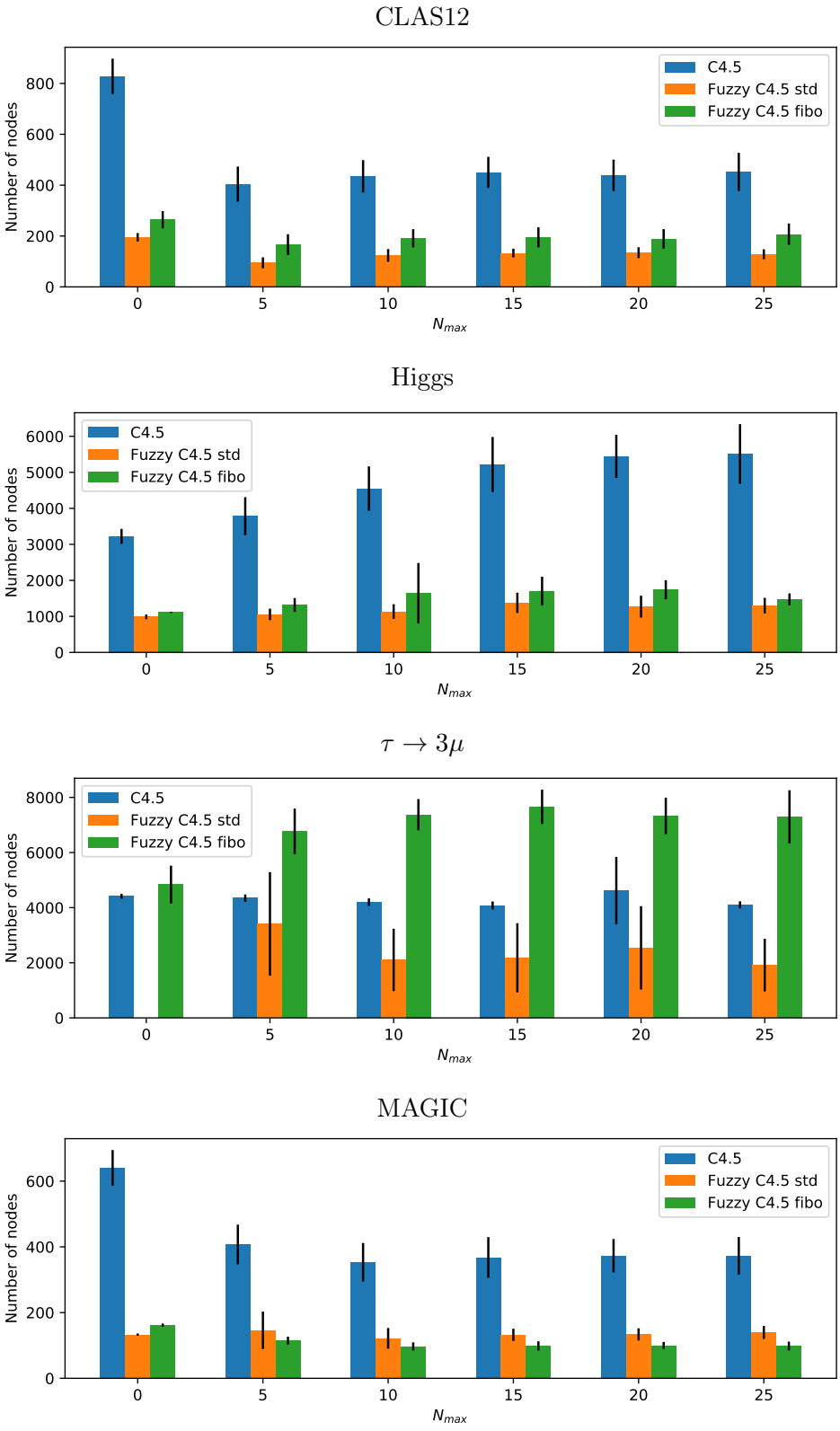


FIGURE 6.4: Number of nodes in trees and standard deviation as function of the number of built features in C4.5 and its fuzzy versions.

6.2 Boosting feature construction in generalized additive models

This section focuses on embedding feature construction in GAM with a few adaptations. An effort is indeed made in this work to include prior knowledge about the *bitonicity* of features. The embedding principle is first detailed in 6.2.1. Then, bitonicity is introduced in 6.2.2, before the global model is proposed in 6.2.3. Experiments are conducted in 6.2.4 and interpretability is discussed in 6.2.5.

6.2.1 Principle

GAM are a sum of univariate shape functions, as seen in 2.2.3. Since each term of the GAM involves a single input feature, feature construction will occur in each term of the GAM in an embedded manner. To be able to build each feature independently, the GAM must be fitted iteratively and not globally. Therefore, gradient boosting seems perfectly adapted to learn the GAM terms.

For a binary classification task, we start from a list of inputs X and a target vector y. A standard classification GAM is modeled by a sigmoid of the sum $F(x)$ of the GAM terms (the link function is $g(\widetilde{y}) = \log \frac{\widetilde{y}}{1-\widetilde{y}}$). In the proposed method, the GAM is built iteratively, building a feature at each step. The objective is to minimize the cross-entropy between the target y and the prediction $p(x) = \frac{1}{1+e^{-F(x)}}$, noted p for conciseness:

$$\mathcal{L}(F(x)) = -\left(y \log(p) + (1-y)\log(1-p)\right) \tag{6.6}$$

We apply a gradient boosting method to improve this loss function (which is convex) at each step of the algorithm. More precisely, the steps to learn a GAM with gradient boosting and embedded feature construction are:

1. Start from a first prediction model predicting a constant $p_0 = \frac{1}{n}\sum_i y_i$ for each x in X. Then, $F_0(x) = \log \frac{p_0}{1-p_0}$.
2. At a given step n, compute the residual r_n:

$$r_n = -\frac{\partial \mathcal{L}(F_{n-1}(x))}{\partial F_{n-1}(x)} = y - p_{n-1}. \tag{6.7}$$

3. Build a new feature z_n using the constrained GP algorithm of chapter 5 as filter method with a fitness function targeting the residual r_n (see precisions below).
4. Fit a term $h_n(z_n)$ to the new feature using any shape function (splines, boosted trees, etc.) and update the GAM: $F_n(x) = F_{n-1}(x) + \alpha_n h_n(z_n)$, z_n being a function of x thanks to the feature construction process. α_n is determined by performing a gradient descent with the goal to minimize the cross-entropy in average. α_n is then directly linked to the importance of the new term in the global result.

Concerning step 3 above, there is no explicit criterion to choose a feature for GAM as opposed to tree-based models. Usually, GAM exploit all available features without specific ordering. However, feature construction requires a specific fitness function. Fitting the GAM term is too time-consuming to be applicable in practice. Instead, we choose to train a shallow univariate decision tree with maximum four leafs on the set $\{(z_i, r_{i,n})\}$, z 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 error between the prediction of the shallow tree and the target r_n.

The method presented here is the basis for GAM with embedded feature construction. In addition, prior knowledge is included about the shape of the distributions of the features.

6.2.2 Bitonicity as prior knowledge

The objective is to integrate prior knowledge during the inference of GAM terms.

For instance, a monotonicity assumption is often made in the literature [Marsala and Petturiti, 2013, Fard et al., 2016a,b, Gupta et al., 2016, Nguyen and Martínez, 2019]: one or several input variables are assumed to be monotonic with respect to the target variable. The machine learning 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, in the HEP field in particular, the most frequently used high-level variables often present a local extremum with respect to the output (see Figure 6.5 for instance). This property is called *unimodality* or bitonicity. It can be used as a constraint in the induction of GAM terms. Bitonicity has been proven to be a relevant constraint, for instance in dose-response analysis [Köllmann et al., 2014]. Bitonicity is introduced here as an extension to monotonicity in the context of HEP applications, to be enforced in GAM terms.

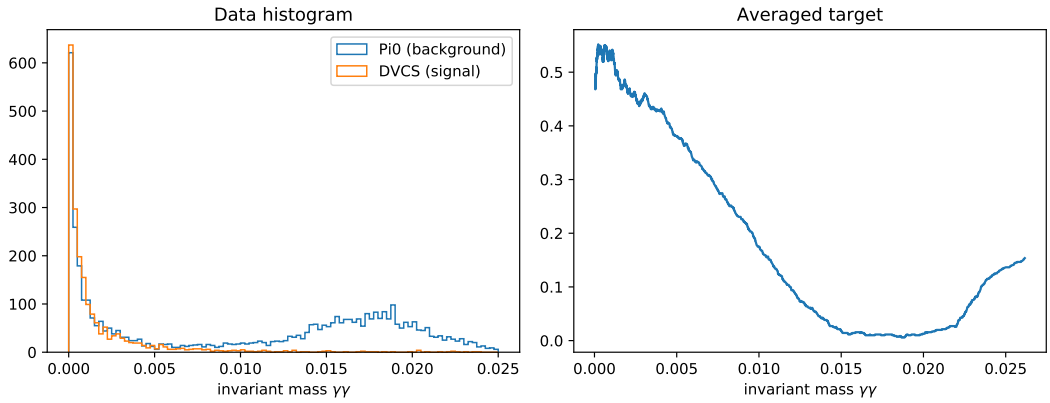


FIGURE 6.5: Invariant mass $\gamma\gamma$, a high-level variable often used in HEP to recognize π^0 production events. On the left, the unnormalized distributions of the two classes with respect to the invariant mass $\gamma\gamma$. On the right, the averaged target (corresponding to the ratio between numbers of signal and background instances per bin).

6.2.2.1 Bitonicity definition

Starting from the definition of monotonicity, we propose a formal definition for bitonicity of a multivariate function.

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 $y \geq x$. Fard et al. [2016a] 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 6 *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 (i.e. binding the end of the function to its start) are here not considered. 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 $D > 1$). Figure 6.6 displays two examples of univariate bitonic functions, one example of a bivariate bitonic function and one bivariate non-bitonic function.

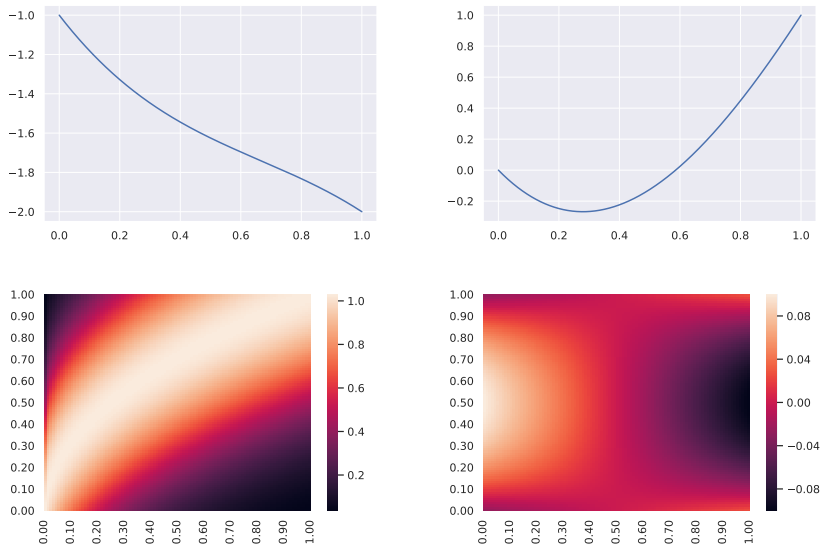


FIGURE 6.6: First two plots: two bitonic univariate functions. Third plot: bivariate bitonic function (i.e. bitonic w.r.t its two variables). Fourth plot: non-bitonic bivariate function, since the y variable is increasing then decreasing for low x and the opposite for high x.

6.2.2.2 Procedure to evaluate the bitonicity degree of a feature

The bitonicity of a function is often hard to prove with an analytical method, except for simple functions [Barthelemy, 2015]. To numerically quantify the *non-bitonicity degree* of a function, it is sampled into an ordered vector (sequence noted v), then compared to a close bitonic sequence.

For instance, 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. This property can be exploited to determine the bitonicity degree of a sequence.

Alternatively, isotonic regression Chakravarti [1989] fits a non-decreasing or non-increasing function to the data. Unimodal regression Stout [2008] extends this idea to unimodal (i.e. bitonic) distributions: the algorithm finds the optimal extremum and fits a non-decreasing function to the left part of the extremum and a non-increasing function to the right part. A method using cumulatives is preferred because it is simpler to implement and since the objective is just to rank features with respect to each other and not to get the closest bitonic approximation.

The proposed algorithm to measure the non-bitonicity of a feature is detailed in Algorithm 3. This algorithm has complexity $O(n^2)$ and involves about $\frac{n(n-1)}{2}$ comparisons by avoiding to recompute all cumulative arrays at each step, with n the size of the sequence. 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 (i.e. its maximum minus its minimum) of the sequence.

Algorithm 3: Evaluation of the non-bitonicity of a sequence

Input : v sequence to check of size n
Output: non-bitonicity degree of v
`// Check monotonicity first`
cummax ← cumulative maximum of v (vector of size n)
cummin ← cumulative minimum of v (vector of size n)
diff1 ← $\sum |v - cummax|$ `// = 0 if` v `is increasing`
diff2 ← $\sum |v - cummin|$ `// = 0 if` v `is decreasing`
minDiff ← min(diff1, diff2)
`// Initialization of the for loop`
cumminBefore ← cummin, cummaxBefore ← cummax
cumminAfter ← cummin, cummaxAfter ← cummax
for $i \leftarrow 0$ **to** $n - 1$ **do**
  if $v[i] > v[i-1]$ **then** `// Recompute cumminAfter`
    cumminAfter[i:] ← cumulative minimum of $v[i:]$
  if $v[i] < v[i-1]$ **then** `// Recompute cummaxAfter`
    cummaxAfter[i:] ← cumulative maximum of $v[i:]$
  diff1 ← $\sum |v[:i] - cummaxBefore[:i]| + \sum |v[i:] - cumminAfter[i:]|$
  `// diff1 = 0 if` v `is increasing then decreasing with maximum at` i
  diff2 ← $\sum |v[:i] - cumminBefore[:i]| + \sum |v[i:] - cummaxAfter[i:]|$
  `// diff2 = 0 if` v `is decreasing then increasing with minimum at` i
  minDiff ← min(minDiff, diff1, diff2)
return $\frac{minDiff}{n(\max(v) - \min(v))}$

A sequence is perfectly bitonic if and only if the application of this algorithm to the sequence returns 0. The non-bitonicity degree of a sequence varies consequently between 0 and 0.5, the worst case being a sequence of alternating zeros and ones.

The bitonicity of a data feature is checked by looking at the variation of the target values along this feature. More precisely, the non-bitonicity of the function $h : y \to h(x)$ is computed, with y the target vector and x the feature. Figure 6.7 illustrates two applications of this procedure.

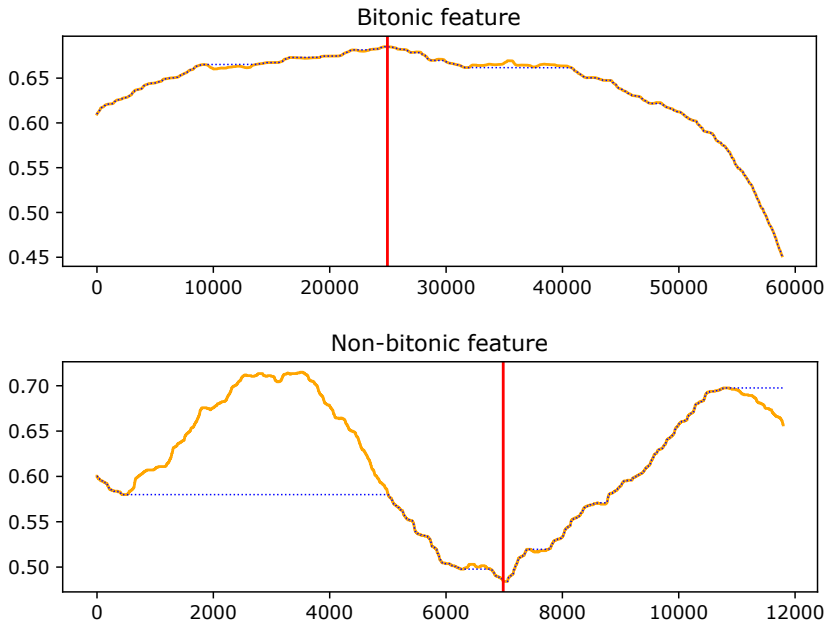


FIGURE 6.7: Two smoothed features (in orange). The y-axis corresponds to the smoothed output value sorted along the tested feature. The x-axis denotes the rank of the sorted output sequence with respect to the tested feature. In dotted blue is represented the cumulative minimum/maximum that is the reference to compute the difference and get the non-bitonicity degree penalty. The red vertical line marks the hypothesis of the algorithm for the extremum. Visually, the first feature is bitonic but not the second. With Algorithm 3, the first feature obtains a non-bitonicity degree of 0.00143 and the second one of 0.14265.

The bitonicity of a two-dimensional array (i.e. of a pair of features with respect to a target variable) can be checked by summing the result of Algorithm 3 over rows and columns.

Evaluating bitonicity in practice

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 $x = (x_1, ..., x_n)$. Average the values of r where x 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. As a consequence of the smoothing, the non-bitonicity degree will rarely reach its theoretical maximum.

6.2.2.3 Using bitonicity as a penalization

During feature construction

Feature construction is embedded in each GAM term fitting during boosting, as detailed in 6.2.1. To enforce bitonicity during feature construction, a bitonicity penalty term is added to the fitness of the candidate feature z_i. This penalty term b is the result of applying the procedure detailed above 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.

In shape functions

As seen in chapter 2, GAM are a sum of univariate shape functions. These shape functions can take different forms, for instance splines or boosted decision trees. 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 sum of squared errors, with a penalization parameterized by a smoothing factor λ. The latter is usually optimized (with respect to a performance metric) by generalized cross-validation (GCV) or restricted maximum likelihood (REML) as seen in chapter 2.

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 above.
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).

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 [Pya and Wood, 2015]. In [Köllmann et al., 2014], 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 as long as the function is not bitonic. Moreover, it can be used with any shape function that supports penalization.

6.2.3 Global GAM fitting method with embedded feature construction and bitonicity

Algorithm 4 summarizes the induction of a complete GAM with (bitonic) shape functions fitted to automatically built (bitonic) features: FCGAM (Feature Construction in Generalized Additive Model).

Algorithm 4: 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
Output: induced FCGAM
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 **do** `// one iteration builds one term of the GAM`
(1) Build one single feature z using feature construction 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 h on the built feature z with target r. Bitonicity may or may not be enforced at this step.
$g \leftarrow +\infty, \quad \alpha \leftarrow \text{random}(0,100)$
while $|g| > \epsilon$ **do** `// gradient descent for` α `with learning rate` β
$\widetilde{F} \leftarrow F + \alpha h, \quad \widetilde{p} \leftarrow \frac{1}{1+e^{-\widetilde{F}}}, \quad g \leftarrow \frac{1}{m}\sum_{i=1}^{m} h_i(\widetilde{p}_i - y_i), \quad \alpha \leftarrow \alpha - \beta g$
$F \leftarrow F + \alpha h, \quad p \leftarrow \frac{1}{1+e^{-F}}, \quad r \leftarrow y - p$
return p

We consider four variants of Algorithm 4 regarding the bitonicity constraints (summarized in Table 6.3). Bitonicity can be enforced or not during feature construction (label `(1)` in Algorithm 4) or for shape functions (label `(2)`). Bitonic FCGAM

(BFCGAM) enforces bitonicity both in features and shape functions.

TABLE 6.3: Four variants of Algorithm 4. 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 feature construction	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

To activate the bitonicity constraint for shape functions for the FCGAM_b_{max} variant, 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} is set at 0.04 (see Figure 6.8).

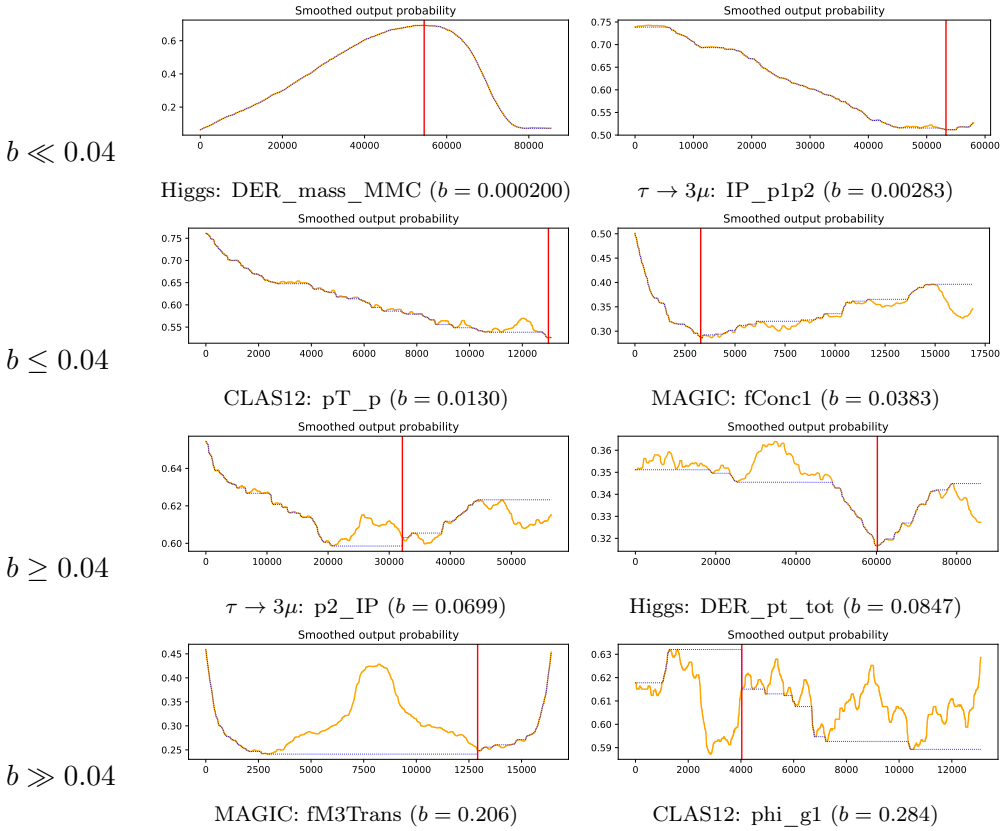


FIGURE 6.8: Feature bitonicity examples on three datasets, from smaller to higher bitonicity penalties. On the graph is plotted the output probability after smoothing (i.e. the vector used for the computation of the bitonicity penalty). In dotted black is the cumulative minimum/maximum that is the reference 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.

6.2.4 Experiments

6.2.4.1 Validation of the principle to enforce bitonicity of shape functions

The method to enforce bitonicity in shape functions is tested on a toy dataset of 1000 instances displayed on Figure 6.9 (data generated with the `make_classification` function of scikit-learn [Pedregosa et al., 2011]). 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 Cohen's kappa score and the bitonicity penalty obtained by the fitted function is plotted on Figure 6.10. Figure 6.11 depicts the fitted terms at small, large, and optimal λ (for which the bitonicity penalty is 0).

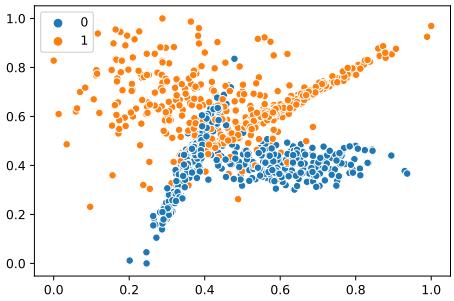


FIGURE 6.9: Toy dataset with two variables x_1 (x-axis) and x_2 (y-axis) and two classes (blue and orange dots).

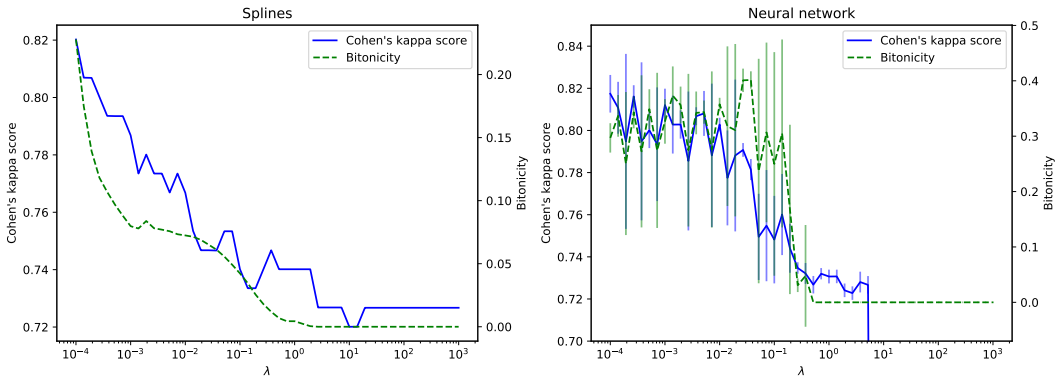


FIGURE 6.10: Evolution of Cohen's kappa score (left y-axis) and function non-bitonicity degree (right y-axis) as function of the regularization parameter λ. Left plot: with splines as shape functions; right plot: with neural networks (two layers of size 100) as shape functions.

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 0.72, the score of the neural network fitted terms finally drops to 0 (outside the scope of the plot) shortly after reaching bitonicity.

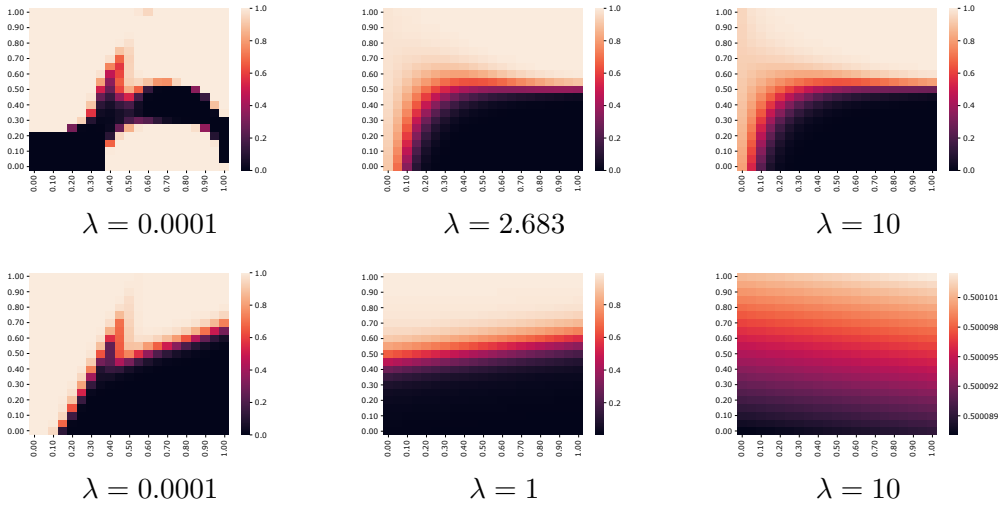


FIGURE 6.11: Top: spline models fitted on toy dataset. $\lambda = 2.683$ corresponds to the minimal λ value for which the bitonicity equals 0. Bottom: MLP models fitted on toy dataset. $\lambda = 1$ corresponds to the minimal λ value for which the bitonicity equals 0.

6.2.4.2 Experimental settings

The four variants presented in Table 6.3 are compared for GAM with splines or neural networks as shape functions.

The same parameters for the constrained GP algorithm for feature construction are kept: 100 generations of 500 individuals.

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.

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 feature construction and shape function fit are done). Boosting is performed for 20 iterations. The convergence is discussed before presenting the main results.

The mean and standard deviation of the Cohen's kappa score are presented for each dataset and algorithm configuration, averaged over at least 5 independent runs and doing 5-fold cross-validation (25 runs in total for each numerical result).

6.2.4.3 Convergence of the boosting algorithm

The global GAM consists of 20 terms, namely 20 shape functions each associated to a built feature. Figure 6.12 displays the evolution of the classification score for each dataset against the number of GAM terms.

Convergence is reached in every case after maximum 20 iterations, whatever the FCGAM variant:

- the maximum needed number of iterations to reach convergence for the CLAS12 dataset is 16, whatever the shape function used or the FCGAM variant;
- for Higgs this number is 18;
- for the $\tau \to 3\mu$ 16;
- and for MAGIC 17.

Convergence is considered achieved if the p-value of a Welch's t-test between the distribution of scores at 20 iterations and the distribution of scores at a sooner iteration is over 0.05, i.e. the two distributions are not significantly different from each other.

In particular, fewer iterations suffice in several cases to reach an already satisfying score, which is sufficiently close to the score reached after 20 iterations. Since interpretability decreases with increasing number of GAM terms, it is interesting to limit this number. This is a trade-off that experts must consider depending on their own criteria.

6.2.4.4 Comparison of the four proposed variants for FCGAM

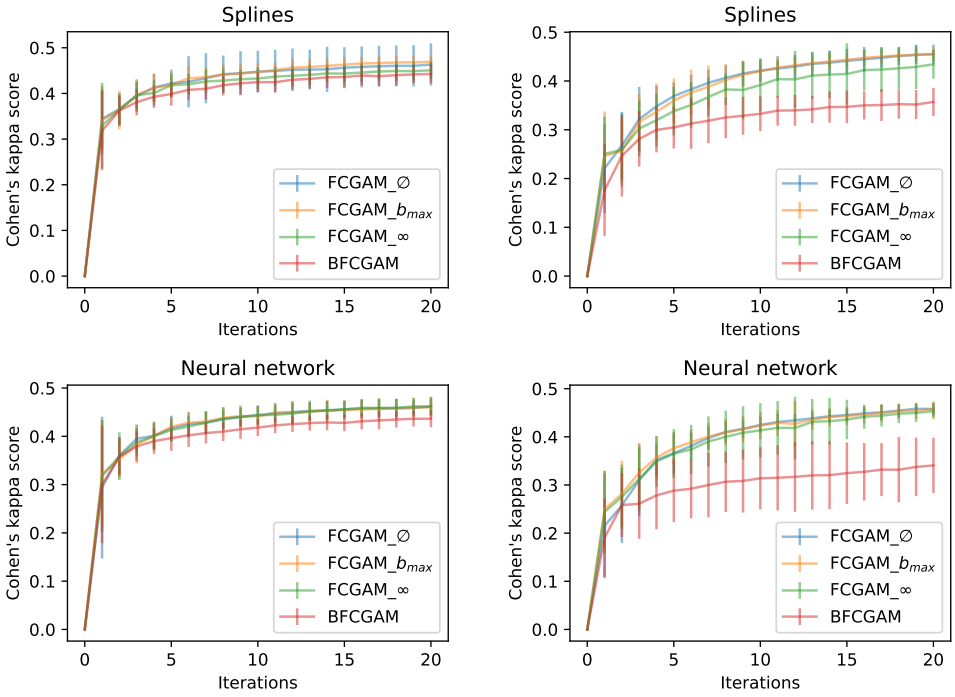
Tables 6.4 and 6.5 presents the results obtained respectively with splines and MLP as shape functions while applying the four variants of the FCGAM algorithm on four datasets.

TABLE 6.4: Cohen's kappa score of the four proposed variants with splines as shape functions.

	CLAS12	Higgs	$\tau \to 3\mu$	MAGIC
GAM	0.320 ± 0.006	0.363 ± 0.007	0.636 ± 0.005	0.668 ± 0.012
FCGAM_$\varnothing$	0.462 ± 0.020	0.458 ± 0.013	0.546 ± 0.017	0.717 ± 0.015
FCGAM_b_{max}	0.460 ± 0.022	0.458 ± 0.013	0.548 ± 0.011	0.715 ± 0.015
FCGAM_∞	0.462 ± 0.017	0.453 ± 0.016	0.548 ± 0.012	0.715 ± 0.016
BFCGAM	0.436 ± 0.017	0.341 ± 0.057	0.446 ± 0.120	0.684 ± 0.015

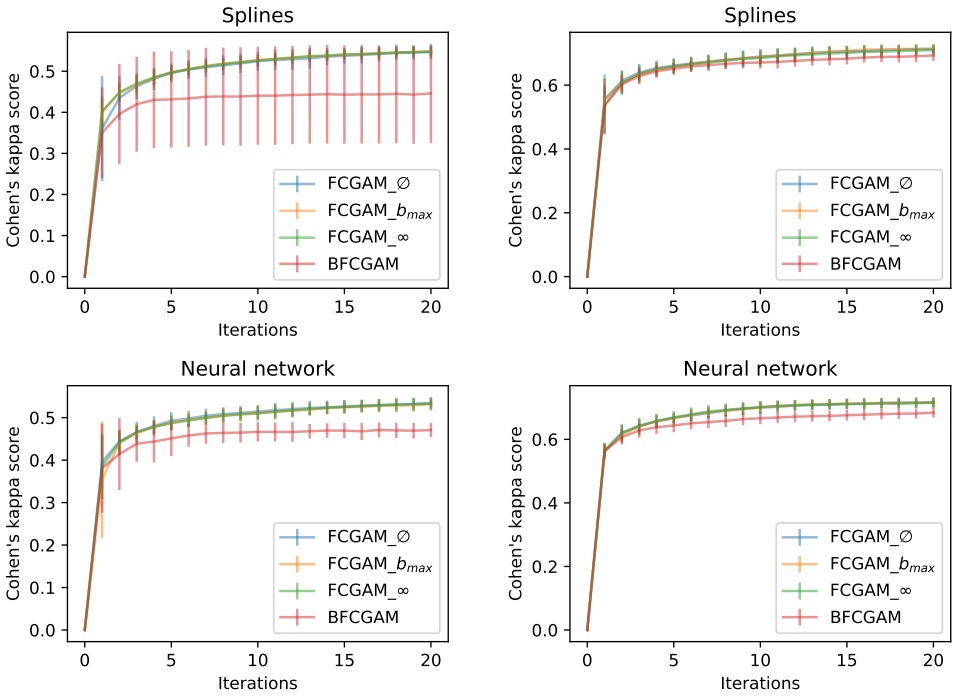
TABLE 6.5: Cohen's kappa score of the four proposed variants with neural networks as shape functions.

	CLAS12	Higgs	$\tau \to 3\mu$	MAGIC
GAM	0.241 ± 0.007	0.329 ± 0.011	0.582 ± 0.006	0.637 ± 0.015
FCGAM_$\varnothing$	0.463 ± 0.046	0.455 ± 0.020	0.535 ± 0.013	0.714 ± 0.015
FCGAM_b_{max}	0.469 ± 0.013	0.455 ± 0.014	0.532 ± 0.014	0.714 ± 0.015
FCGAM_∞	0.451 ± 0.019	0.434 ± 0.029	0.532 ± 0.015	0.709 ± 0.016
BFCGAM	0.443 ± 0.020	0.357 ± 0.029	0.471 ± 0.017	0.693 ± 0.017



(A) CLAS12 dataset

(B) Higgs dataset



(C) $\tau \to 3\mu$ dataset

(D) MAGIC dataset

FIGURE 6.12: Evolution of the Cohen's kappa score as function of the number of boosting iterations in FCGAM, for the four datasets and the two shape functions (splines on the top, MLP on the bottom).

A first observation is that feature construction permits to improve significantly the classification scores except for the $\tau \to 3\mu$ dataset. Contrary to previous experiments, here the model only uses newly built features and discards the original ones. Since we observed that $\tau \to 3\mu$ does not benefit much from feature construction, it is not surprising that the classification performances degrade for this dataset if the original features are removed.

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 and $\tau \to 3\mu$ datasets in particular. In all cases, letting the feature construction free and enforcing bitonicity 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, but can be too restrictive: some really discriminative features are not bitonic and are indispensable to get a good score. This is probably the case for the Higgs dataset.

6.2.4.5 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 6.13 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 feature construction 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.

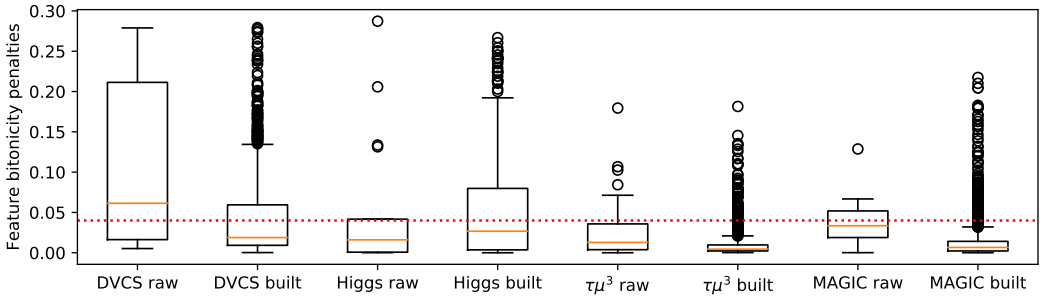


FIGURE 6.13: Boxplots of features non-bitonicity degrees. The box indicates the first and third quantiles with an orange line at the median, while the whiskers extend to the farthest data point within 1.5 IQR (interquartile range) after the box. The horizontal dotted red line represents the bitonicity threshold b_{max} set to 0.04 to trigger the bitonicity constraint on the shape function.

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 bitonicity. Indeed, the Higgs and $\tau \to 3\mu$ datasets present mainly bitonic raw features whereas their scores are impaired under bitonicity constraints, when it is the opposite for CLAS12.

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 CLAS12 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.

$\tau \to 3\mu$ is a particular case. Indeed, this dataset comprises several ordinal features, i.e. features taking a limited number of values but that are still orderable real numbers and usable in feature construction. The bitonicity of built features combining continuous raw features with ordinal features is more difficult to evaluate. Notably, the dataset comprises seven variables that take the value 0 from 27 to 77% of times. Therefore, combinations of these variables with continuous variables will have a large number of instances at 0. Since the bitonicity penalty computation depends on the variable distribution, it will be biased by the zero-valued instances. The bitonicity penalty will be driven down closer to 0 because of this particularity of the feature. In addition, such features are poorly suited for fitting shape functions.

6.2.4.6 Impact of the bitonicity threshold

The parameter b_{max} for the application of the bitonicity constraint on shape functions has been determined experimentally by looking at raw features of the considered datasets. This part aims at evaluating the impact of this b_{max} parameter on the classification performance of the FCGAM_b_{max} variant.

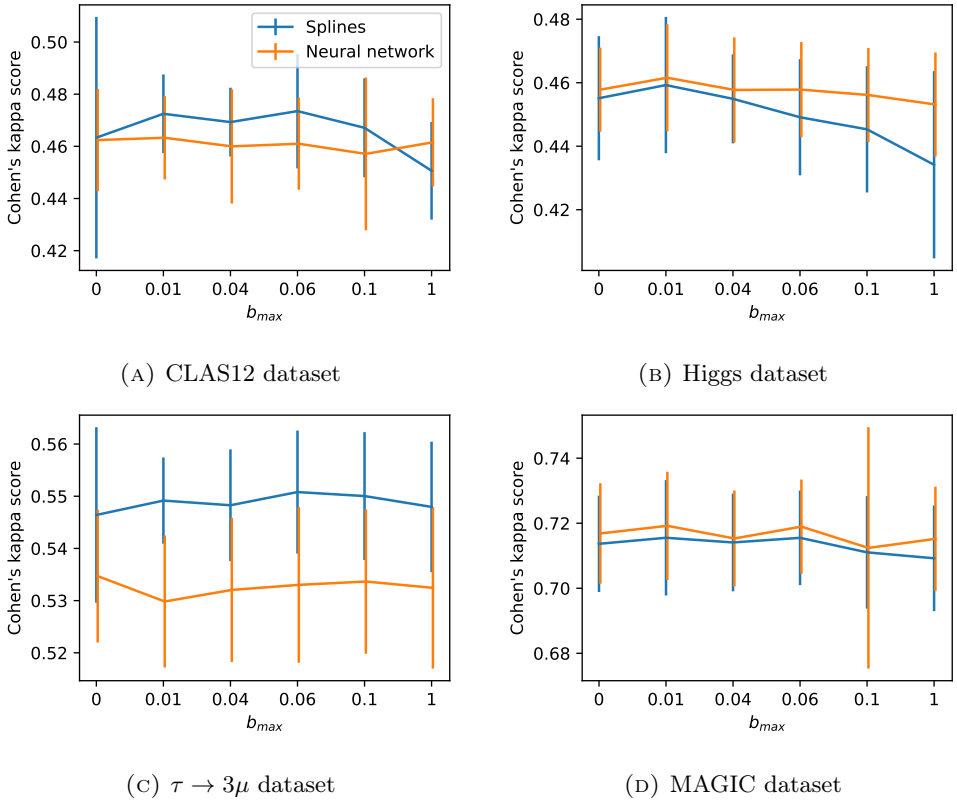
The evolution of the Cohen's kappa score against the non-bitonicity threshold b_{max} is displayed on Figure 6.14 for each dataset and shape function. In the neighborhood of $b_{max} = 0.04$, no significant variation in the Cohen's kappa score is observed. Since $b_{max} = 1$ corresponds to the FCGAM_∞ variant, the classification scores indeed drops significantly for the Higgs dataset.

In conclusion, modifying b_{max} around the determined value 0.04 (for instance 0.01 or 0.06) does not alter significantly the classification performances. The latter start dropping from $b_{max} = 0.1$ for some datasets. The presented results for FCGAM_b_{max} with $b_{max} = 0.04$ are therefore stable.

6.2.5 Discussion on interpretability

The global GAM models consist of 20 terms, namely 20 shape functions each associated to a built feature. As discussed with Figure 6.12, a choice must be made between classification performance (increasing with the number of terms) and interpretability (decreasing with complexity hence decreasing with the number of terms).

We analyze now a few features and the associated fitted shape functions for the CLAS12 and Higgs dataset. Figures 6.15, 6.16 and 6.17 are presented in the same way: the left plot is the target vector binned along 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



(A) CLAS12 dataset

(B) Higgs dataset

(C) $\tau \to 3\mu$ dataset

(D) MAGIC dataset

FIGURE 6.14: Cohen's kappa score against bitonicity threshold b_{max} for the four datasets. $b_{max} = 0$ is equivalent to the FCGAM_$\varnothing$ version of the algorithm, and $b_{max} = 1$ to FCGAM_∞.

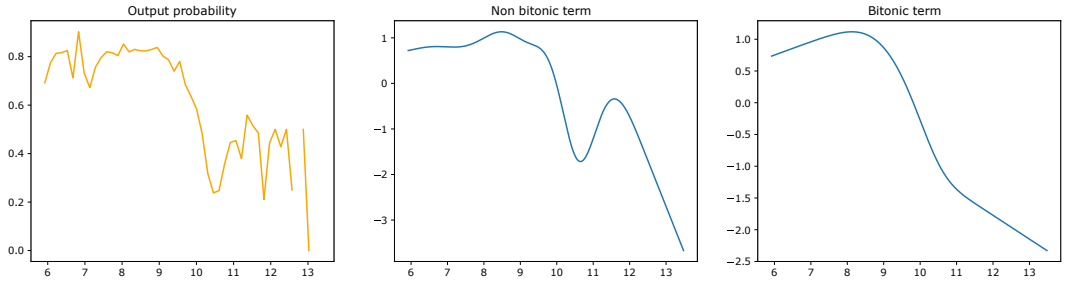


FIGURE 6.15: CLAS12: $p_e^z + p_p^z + p_{\gamma_1}^z$. Lower value means higher probability to have a signal event.

for this feature using splines without bitonicity enforcement, and the right plot is a GAM term learnt using splines and with bitonicity enforcement.

Figure 6.15 illustrates a feature for the CLAS12 dataset, which is the sum of the z momenta that has already been encountered in previous experiments on feature construction. 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/c. If the sum of the momenta of the output particles is inferior to 10.6 GeV/c, additional particles have been produced from the collision and hence it is not a signal event. This explanation is well reproduced in the non-bitonic term on the central plot, but disappears in the bitonic term on the right plot. In this particular case, enforcing bitonicity is not a good idea as the important thing is the peak at 10.6 GeV/c.

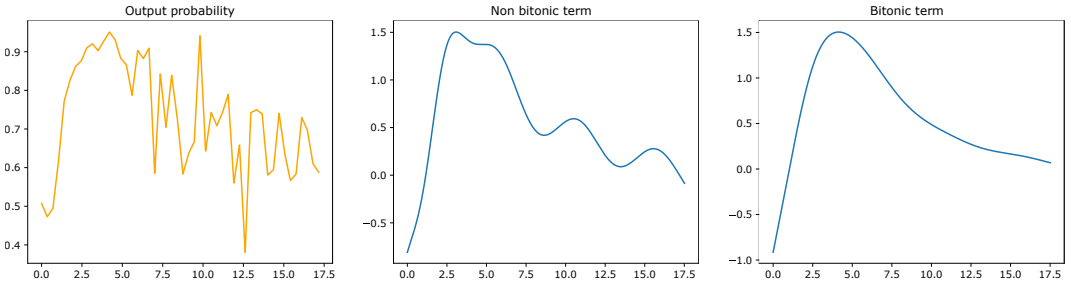


FIGURE 6.16: CLAS12: $\angle(p_{\gamma_2}, p_{\gamma_1} + p_{\gamma_2})$. Lower value means higher probability to have a signal event.

Figure 6.16 illustrates a CLAS12 built feature, which is the angle between γ_2 the second highest energetic photon and the sum of two detected photons $\gamma_1 + \gamma_2$. A signal DVCS event involves a single γ photon. Nevertheless, a second uncorrelated photon from background may be simultaneously detected. It then resembles the major background being π^0 production. The two γ photons of this background process are produced by the decay of a same particle, the π^0, and are therefore correlated due to the energy-momentum conservation. Consequently, the distribution of this angle is not random and presents a peak around 5 degrees. However, the oscillations learnt in the non-bitonic term are probably learnt from the noise present in the data distribution. The bitonic term permits to solve this irregularity: experts can visually tell that it generalizes better and is more consistent with their expectations.

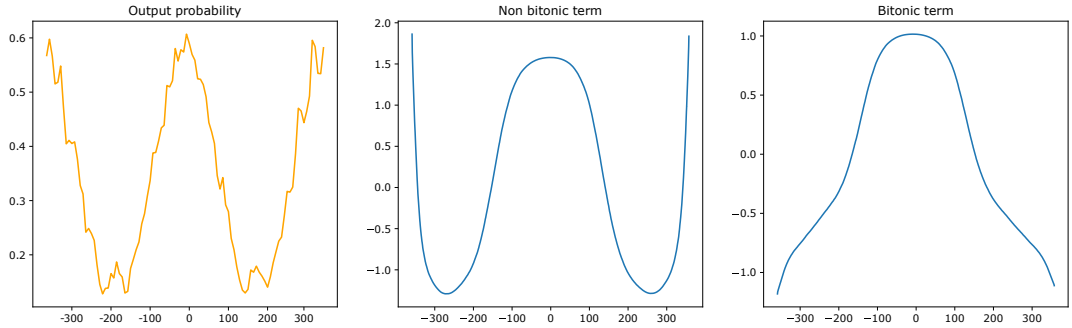


FIGURE 6.17: Higgs dataset: $\phi_{lep} - \phi_{missingtE}$. Higher value means higher probability to have a signal event.

Another case of bitonicity issue is illustrated by Figure 6.17. The angle between the lepton and a hypothetical missing particle to enforce momentum conservation of the collision is one of the most common feature built by a FCGAM for the Higgs dataset. Indeed, this missing momentum actually relates to neutrinos, which are undetectable particles. In signal events, the neutrinos are in majority emitted in the same direction than the lepton. However, in several background processes, there is only one neutrino emitted in the opposite direction of the lepton (see Figure 6.18). Therefore, the probability to have a signal event is higher at 0 degrees and at its lowest at -180 and 180 degrees. This particular feature is highly discriminative but not bitonic. Enforcing bitonicity on this feature would be counterproductive. However, this particular case could be solved in two different ways:

- the feature construction could have found a bitonic equivalent for this feature, for instance by computing its absolute value or its square;

- we could authorize more than one shape function to fit the same feature, but in this case we would lose in interpretability.

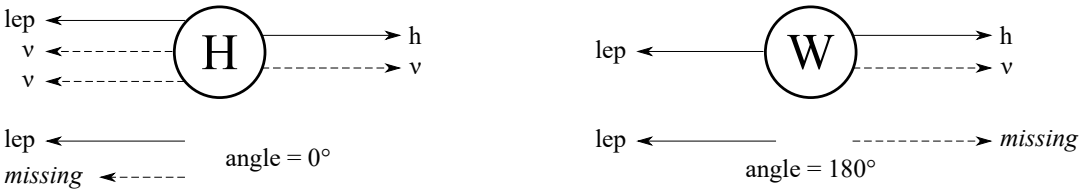


FIGURE 6.18: Illustration of signal events on the left and one type of background events on the right.

6.3 Conclusion and perspectives

This chapter proposed different methods to embed the constrained feature construction methods introduced in chapter 5 in several transparent models. First, embedded feature construction in tree-based and rule-based models has been studied: it improves the classification performances (up to 0.260 for FURIA with the CLAS12 dataset) while limiting the overall dimensionality of the models, and producing only interpretable features. The proposed method can be applied without loss of generality to other tree-based or rule-based models that were not included in the experiments of this chapter. Secondly, we proposed to embed the feature construction technique into GAM. Again, it significantly improves the classification scores (with a gain for CLAS12 of 0.142), while producing interpretable features for physicists. In addition, we introduced bitonicity 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. The combination of feature construction with bitonicity enforcement increases the interpretability potential and generalization power of the global model, with a classification score comparable to the score obtained without constraint, if not greater. Bitonicity extends the notion of monotonicity in the literature, and we can imagine several applications where it can be useful to constrain models to enhance interpretability.

Further work can be carried out to understand why bitonicity is more compliant with some datasets than others. In addition, a very interesting extension of this work could be conducted using bivariate terms as part of a GA^{2}M model. This would imply the embedded feature construction of a pair of features for each bivariate term, as well as an extension of the bitonicity evaluation procedure and constraint enforcement.

Conclusion of part II

Chapter 4 introduced the field of feature construction with a review of the state of the art. The feature construction methods mainly divide into tree-based and evolutionary-based techniques, the latter being the most widespread. In particular, genetic programming is a very flexible tool for feature construction. Notably, several techniques exist to constrain GP. Besides, three approaches coexist to perform feature construction:

- filter fitness functions can be used that are faster but lead to less specialized features;
- wrapper fitness functions give the best discriminative power;
- embedded feature construction exploits most of the times filter fitness functions but embeds them into the induction of machine learning models instead of performing feature construction ahead of it.

Therefore, drawn from the state of the art, chapter 5 proposed a constrained GP algorithm for feature construction adapted to experimental physics analyses. A grammar is designed so that only valid feature combinations are made, and a transition matrix is proposed to guide the feature search. Experiments on prior feature construction with a filter fitness function and two wrapper fitness functions confirm the significant contribution of the proposed feature construction techniques to improve the classification score (up to 0.260 for FURIA with embedded feature construction with the CLAS12 dataset) while obtaining a superior interpretability compared objectively to other feature construction methods (PSO for prior feature construction, Monte Carlo search for embedded feature construction in tree-based models).

Chapter 6 deepened the experiments with embedded feature construction in tree-based and rule-based models as well as in GAM. For the latter, bitonicity constraints have been introduced to further strengthen interpretability. Embedded feature construction allows to build more features in a given amount of time than prior feature construction, although the produced features are more specific and less interpretable especially in deeper nodes of decision trees.

Table 6.6 summarizes the results obtained for the CLAS12 dataset. Globally, the models with embedded feature construction outperform their equivalent with prior feature construction, at the expense of more built features. The fuzzy decision trees and FURIA with embedded feature construction are the best models in terms of classification performance.

Regarding interpretability, features built through prior feature construction are probably more general than those built in an embedded way. Moreover, rule bases are smaller when induced with prior feature construction compared to embedded feature construction. However, it takes more computation time to build features with wrapper

TABLE 6.6: Summary of the classification performances of the proposed models for the CLAS12 dataset without missing values. FC stands for feature construction.

	Cohen's kappa
Prior FC (5 features, CART as fitness and evaluator)	0.362 ± 0.020
Prior FC (5 features, CART as fitness and C4.5 as evaluator)	0.445 ± 0.021
Prior FC (5 features, FURIA as fitness and evaluator)	0.459 ± 0.010
Embedded FC (C4.5, 15 features)	0.476 ± 0.025
Embedded FC (fuzzy C4.5 std, 15 features)	**0.492 ± 0.021**
Embedded FC (fuzzy C4.5 Fibo, 15 features)	**0.496 ± 0.026**
Embedded FC (CART, 100 features)	0.437 ± 0.044
Embedded FC (AdaBoost, 50 features)	0.447 ± 0.015
Embedded FC (GradientBoosting, 350 features)	0.482 ± 0.031
Embedded FC (FURIA, 2 features per rule)	**0.496 ± 0.039**
Embedded FC (FCGAM_b_{max} splines, 16 terms)	0.466 ± 0.014

fitness functions such as CART or FURIA compared to embedded feature construction, where the fitness function is an information metric. The GAM could not be used as a fitness function for prior feature construction for this reason.

The models implemented in scikit-learn, namely CART, AdaBoost and GradientBoosting are discarded for CLAS12 data analysis in the remaining of this thesis because they do not handle missing values.

Finally, we will consider three of these models to go further in the CLAS12 data analysis in the next part. In particular, we consider FURIA with prior feature construction of 5 features, fuzzy C4.5 Fibo with 15 features built in an embedded way, and FCGAM with 16 terms with splines as shape function and bitonicity enforcement in shape functions for bitonic features.

The classification performances of these models on the CLAS12 dataset containing missing values are printed in Table 6.7.

TABLE 6.7: Classification performances of the retained models on the CLAS12 dataset with missing values. FC stands for feature construction.

	Cohen's kappa
Prior FC (5 features, FURIA as fitness and evaluator)	0.361 ± 0.021
Embedded FC (fuzzy C4.5 Fibo, 15 features)	0.463 ± 0.018
Embedded FC (FCGAM_b_{max} splines, 16 terms)	0.315 ± 0.014

Part III will tackle the transition from the induction using simulated data to the application to real CLAS12 data.

Part III

From simulation to real CLAS12 data analysis

Introduction

This part tackles more specifically the DVCS event selection at CLAS12, and the transition from simulated data to real CLAS12 data. Indeed, transparent models with automated feature construction have been developed in part II, tailored to HEP datasets such as simulation data provided for CLAS12. We state that these models are both interpretable and efficient for event selection, and that they can be used for DVCS analysis from CLAS12 data. These two assertions will be verified in this part. However, a problem remains, which is the distribution shift between simulation and real data. Due to uncertainties in the experimental setup and in theoretical physics models, the data used for training may not reflect perfectly the reality. In addition, the interpretability of the proposed models must be validated by the physicists users themselves, so that they can be effectively be employed for a physics analysis. Finally, the performances of these models for the actual physics analysis must be demonstrated.

Using the proposed adapted transparent machine learning models with prior or embedded feature construction, the objectives are now to assess their performances in terms of classification on real data and interpretability on the one hand, and to perform the final analysis of DVCS data on the other hand.

To these ends, three chapters constitute this part:

- Chapter 7 introduces *domain adaptation* to handle the distribution shift between simulated data and real data. The objective is to better evaluate the classification performances of the models on real data and if possible to improve them.
- Chapter 8 conducts a survey to evaluate the perceived interpretability of the proposed models by a population of experimental physicists. This study should permit to verify whether these models could be used in practice for an analysis.
- Chapter 9 applies the proposed models on real data, with the objective to extract the DVCS asymmetry and compare the results with those obtained by a standard analysis.

Chapter 7

Model transfer to real data

7.1 State of the art of domain adaptation 137
7.1.1 Domain adaptation via invariant feature representation . . 137
7.1.2 Domain adaptation via domain mapping 139
7.2 Domain adaptation of the particles momenta 142
7.2.1 Constitution of a training dataset: control process 142
7.2.2 Proposed method for domain adaptation from simulation to CLAS12 data 145
7.3 Experiments 146
7.3.1 Experimental settings 146
7.3.2 Experiments with smeared simulated data with flat distributions 149
7.3.3 Experiments with smeared simulated data with cross-sections 152
7.3.4 Domain adaptation to real data 154
7.3.5 Adapting transparent models to transported data 156
7.4 Conclusion and perspectives 159

While all previously presented machine learning models have been trained on simulated data, there exist some differences in the distributions between simulated and real data from CLAS12. Indeed, several sources of uncertainties coexist:

- In case of using a Monte Carlo simulation considering the cross-sections, the latter are not perfectly known and therefore may be quite far from reality: they are actually the quantities physicists want to measure;
- The detector responses are imprecise. Indeed, the detectors themselves can be partially inefficient due to imperfections or environmental factors not taken into account in the simulation;
- Contrary to simulation, the real-world detector geometry is not perfectly known, which induces imprecisions during event reconstruction.

In total, putting aside the errors on cross-sections, the real data present biases due to distortions of the phase space and loss of acceptance.

Therefore, the models trained on simulation data are not optimal when applied to real data. First, some of the variables are noisier than expected. For instance, the missing mass $ep \rightarrow e\gamma X$ is shifted and broader in real data compared to simulated data, as displayed on Figure 7.1.

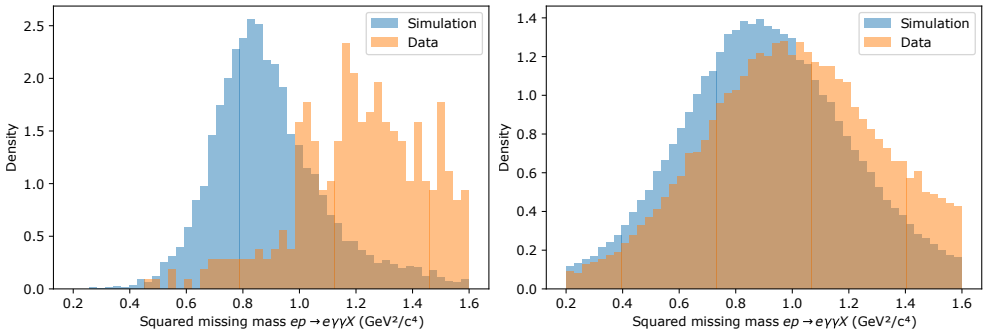


FIGURE 7.1: Squared missing mass $ep \to e\gamma\gamma X$ for exclusive π^0 production events in Monte Carlo simulated data and in CLAS12 data. Left: photons were sent in the forward tagger ($\theta < 5°$); right: in the forward detector ($\theta > 5°$).

These remarks require either to find variables that are less affected by the change of distributions or to shift split thresholds in decision trees and rule bases for instance. Secondly and most importantly, the estimation of the model classification performance is wrong, since this estimation is made on labeled data as well. However, an accurate estimation of the percentage of selected DVCS events as well as of the percentage of contaminating π^0 production events is crucial for the analysis.

Transfer learning is the field of machine learning that consists in adapting a model trained on a source task to be applied to a somehow related target task. In this chapter, a method is proposed to overcome the distribution shift between simulated and real data to get a more accurate evaluation of performances of the machine learning models.

In transfer learning, different sub-fields exist depending on the nature of the differences between the source and target domains [Kouw and Loog, 2019]: the task might be different, the input features as well, the input and/or target value distributions might change between the source and target domains.

However, the majority of techniques of transfer learning suppose to have a few labeled data in the target domain. This is not exactly possible in our case:

- It is possible to obtain a quasi-pure DVCS sample but only in a reduced region of the phase space where the π^0 production cross-section is much smaller than the DVCS cross-section. Therefore, the labeled data in the target domain that could be obtained would not be representative.
- The simulation on which the models are trained comprises exclusively DVCS and π^0 production events. However, numerous other processes occur that are present in real data, not to mention a constant background noise difficult to simulate.

Therefore, transferring trained models is not directly possible. Instead, we focus on *domain adaptation*, which is a sub-field of transfer learning aiming at learning a transformation between the source domain and the target domain. The objective is to learn a transformation of CLAS12 simulated data so that they reproduce better the real distributions. The machine learning models can then be either transferred to the transformed data or entirely retrained.

Section 7.1 presents a summary of the state of the art for domain adaptation, and then our method adapted to domain adaptation of CLAS12 data is detailed in section 7.2.

7.1 State of the art of domain adaptation

Domain adaptation is the sub-field of transfer learning where no labeled target data is available, the input domains differ between source and target and the task remains the same. Surveys can be found in [Kouw and Loog, 2019], [Wilson and Cook, 2020] or [Farahani et al., 2020]. Three types of distribution shifts between source S and target T are covered in the literature:

- *Prior shift*, where the prior distributions of classes y changes ($p_S(y) \neq p_T(y)$ with S and T the source and target domains) but the conditional probabilities of data x given class y are equal ($p_S(x|y) = p_T(x|y)$). This means that the prior assumption on the relative proportion of the different classes is wrong.
- *Covariate shift*, where the distributions of examples are different ($p_S(x) \neq p_T(x)$) but the posterior distribution of classes y remains the same ($p_S(y|x) = p_T(y|x)$). This shift is the most widely tackled in the literature.
- *Concept shift*, where the prior distributions of examples are equal between source and target, but the posterior distributions of classes are different: $p_S(x) = p_T(x)$ but $p_S(y|x) \neq p_T(y|x)$.

Solving prior or concept shifts requires to have labeled data from both source and target domains. Besides, covariate shift can be tackled in an unsupervised manner, although conditional methods exist to take into account label information.

The remaining of this review focuses on unsupervised domain adaptation where neither label information is available on source nor on target data. Therefore, only the covariate shift can be tackled. When the source and target domains are sufficiently similar, instance-based domain adaptation methods can be applied. They often consist in sample reweighting so that the distribution of weighted source samples resembles the target distribution [Sugiyama et al., 2008, Gretton et al., 2009]. When the shift is too substantial, notably when the supports of the source and target distributions are disjointed for instance, reweighting is not efficient. Feature-based domain adaptation methods are therefore more appropriate. We divide these methods in two categories: those that seek a feature representation that is invariant of the domain on the one hand and those that build a mapping between the source and target domains on the other hand.

7.1.1 Domain adaptation via invariant feature representation

While covariate shift implies that the input distributions vary between the source and target domains, finding a feature representation that is invariant with respect to the domain permits to train a general classifier on this representation. The idea of finding an invariant feature representation is validated by the theory of Ben-David et al. [2007, 2010]: the transferable information across domains should not comprise any domain-specific information and therefore should be independent of the domain. This representation search can be performed ahead of the classifier training or simultaneously with it.

Among this first category, Pan et al. [2010] propose transfer component analysis to minimize the maximum mean discrepancy (MMD) [Gretton et al., 2012] metric between representations of the source and target domains. The MMD is a distance between distributions: more precisely, it is the difference between moments of the distributions. MMD is defined by a feature map φ (linked to the so-called kernel through

a scalar product) to compute these moments. Then, the MMD between distributions P and Q writes:

$$\mathrm{MMD}(P, Q) = \|\mathbb{E}_{X\sim P}[\varphi(X)] - \mathbb{E}_{X'\sim Q}[\varphi(X')]\|_{\mathcal{H}} \tag{7.1}$$

where $\mathcal{H}$ is the output space of φ, a so-called reproducing kernel Hilbert space. The choice of the kernel permits to choose which "moments" of the distributions to compare. Joint domain adaptation [Long et al., 2013] extends the idea of Pan et al. [2010] by taking into account the conditional distributions. Later with the development of deep learning, many methods appeared based on autoencoders: the objective of these networks is to encode the input into a lower-dimensional representation while being able to reconstruct the original. Glorot et al. [2011] and Chen et al. [2012] notably use stacked denoising autoencoders to extract invariant features. A general classifier is trained afterwards on these invariant features. Ghifary et al. [2016] use a similar idea with deep reconstruction-classification networks.

Recent methods simultaneously perform invariant feature extraction and classifier training. Indeed, deep learning is known for its ability to learn relevant feature representations [Bengio et al., 2013]. Long et al. [2015] were the first to propose a deep adaptation network (DAN): while the first layers are common to all domains, task-specific layers align the distributions across domains using multiple kernel variants of MMD. Ganin et al. [2016] propose a domain adversarial neural network (DANN) to perform both classification and feature extraction: first layers are common and then the network divides into two branches. The first one classifies the instances and has a classical cross-entropy loss, while the second branch is a domain classifier that verifies whether it is possible to recognize source examples from target examples. The loss of this second branch is a reversed cross-entropy so that the common layers are trained to build an invariant feature representation. The DANN principle is illustrated on Figure 7.2. Tzeng et al. [2015] also use an adversarial architecture and takes into account both marginal and conditional distributions. Deep CORAL [Sun and Saenko, 2016] uses the principle of correlation alignment of [Sun et al., 2015] and embeds it into a deep architecture.

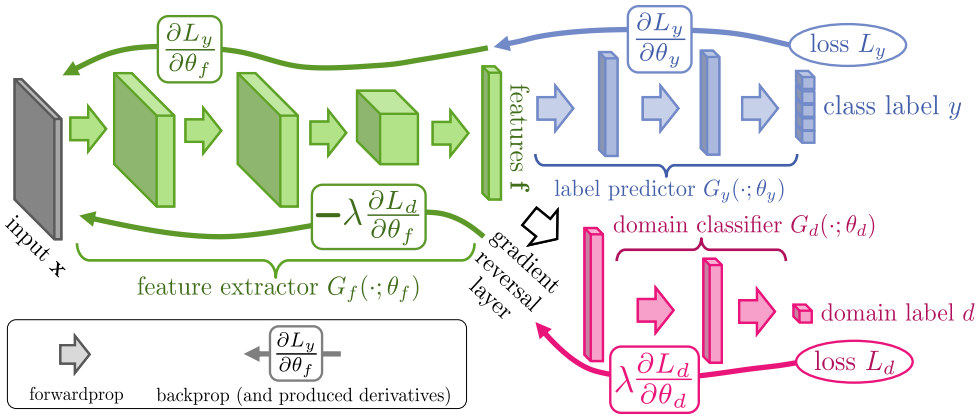


FIGURE 7.2: Domain-adversarial neural network [Ganin et al., 2016].

However, not all methods presented in this first category are compliant with interpretability: the invariant feature representation loses all pre-existing information about the original input features.

7.1.2 Domain adaptation via domain mapping

A second category of feature-based domain adaptation techniques performs a mapping from source to target domains instead of finding a common feature representation.

A simple idea consists in renormalizing the features: Wang et al. [2019] replace the batch normalization in neural networks by a "TransNorm" layer that uses domain-specific mean and variance to improve the network's generalization capacity. CORAL [Sun et al., 2015] aligns second-order statistics in addition to the means.

A few methods project data on a latent space and then find a transformation between the source and target latent spaces. Principal component analysis (PCA) can for instance be used to project data into a lower-dimensional space. Gopalan et al. [2011] then find a geodesic path between the source and target points. Gong et al. [2012] extends their approach with a kernel-based domain adaptation. Finally, Chopra et al. [2013] perform the transition between projected source and target domains with a sequence of autoencoders. The problem with these projection-based methods is that the transformation is performed on the projections, therefore interpretability is lost again.

Direct mapping techniques transform directly the source input space. The two main methods are optimal transport and deep learning.

7.1.2.1 Optimal transport

The problem of optimal transport (OT) has been introduced by Monge [1781]. The problem is to find a solution of least effort to transport mass from one distribution to another. More rigorously, OT is the search of a transport map T from the source space Ω_S to the target space Ω_T that minimizes the transportation cost $C(T)$:

$$C(T) = \int_{\Omega_S} c(X, T(X)) dP(X), \tag{7.2}$$

where P is the marginal distribution over Ω_S, and c a distance. Then, the optimal transport map T^* verifies:

$$T^* = \underset{T}{\operatorname{argmin}}\ C(T) \text{ so that } T\#P = Q \tag{7.3}$$

where $T\#P$ means the transformation of distribution P by the map T and Q is the marginal distribution over Ω_T.

This transport map defines then a distance across distributions called the Wasserstein distance of order p:

$$W_p(P, Q) = \left(\int_{\Omega_S} c(X, T^*(X))^p dP(X) \right)^{\frac{1}{p}}. \tag{7.4}$$

In the particular case where c is the Euclidean distance, W_1 is called the Earth-Mover distance. It has been applied early to measure similarity between images [Rubner et al., 2000].

A minimizer of equation (7.3) does not exist in the general case. The simpler example is the one-dimensional case, where $T^*(X) = \mathcal{C}_T^{-1}(\mathcal{C}_S(X))$ with $\mathcal{C}_S$ and $\mathcal{C}_T$ the cumulative distribution functions of respectively the source and target domain. This

particular case is illustrated on Figure 7.3. In the general case, relaxations are applied so that the problem is not intractable and a suboptimal transport plan can be found.

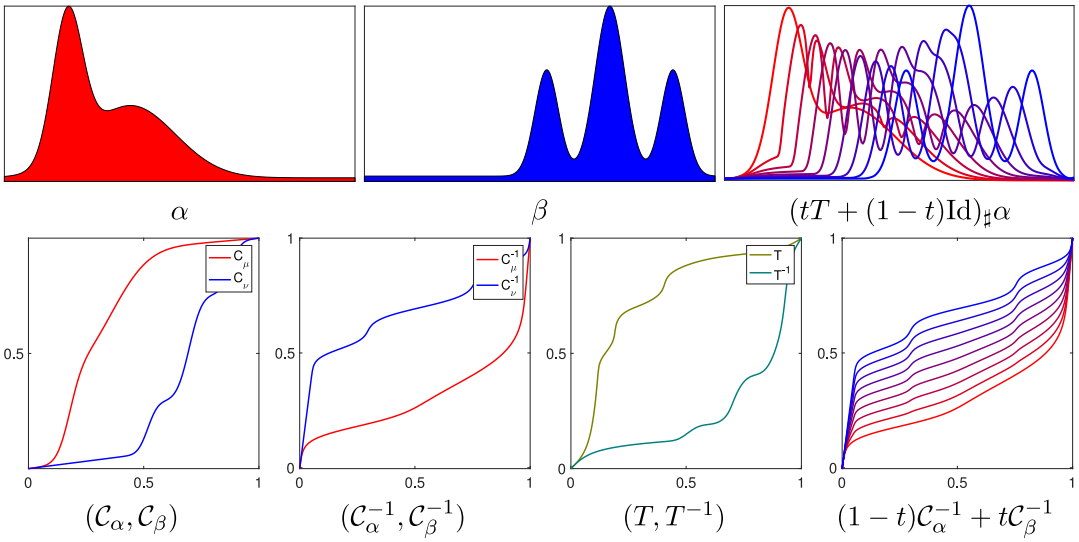


FIGURE 7.3: Illustration of optimal transport in the one-dimensional case [Peyré and Cuturi, 2019]. Source and target distributions P and Q are noted α and β on the figure.

In domain adaptation, only samples of P and Q are available and not the explicit distributions. Optimal transport between two points clouds is solvable but costly [Kuhn, 1955]: the complexity is $O(n^3)$, which delayed the introduction of OT in machine learning. Nearest-neighbors are used to transport new samples.

Cuturi [2013] however proposes an entropic regularization to the optimization problem of OT: the resulting problem is strictly convex and can be solved easily. OT started being used in machine learning notably by Courty et al. [2016] or by Redko et al. [2019] and Turrisi et al. [2020] for multiple sources. The principle is to map the labeled source domain to an unlabeled target domain, as illustrated on Figure 7.4. Other regularizations are possible depending on the nature of the data, for instance Laplacian for graphs [Courty et al., 2016], or using the known classes in the source domain [Courty et al., 2016]. Joint distributions can be used instead of marginal distributions as in [Courty et al., 2017].

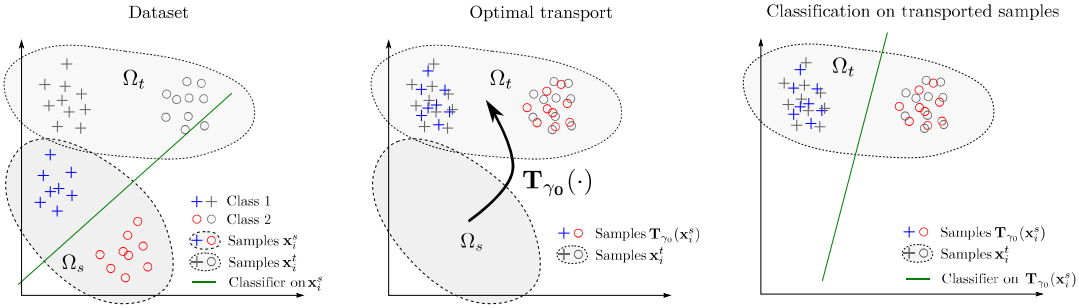


FIGURE 7.4: Illustration of the use of optimal transport in machine learning [Courty et al., 2016]. The classifier is trained on the transported samples.

Finally, there are recent proposals of combinations between OT and deep learning, notably using the Earth-Mover distance [Bhushan Damodaran et al., 2018, Chen et al.,

2018]. The most notable contribution of OT to deep learning is found in the work of [Arjovsky et al., 2017]: a Wasserstein generative adversarial network (Wasserstein GAN) permits to generate samples following a given distribution.

The principle of a GAN is the competition between two networks [Goodfellow et al., 2014]: a generative network whose objective is to generate samples as "realistic" as possible, and a discriminative network that takes as input both real samples drawn from the target distribution and samples produced by the generator. The two parts are trained in alternation. The discriminator is trained to classify real samples from fake generator-produced samples, while the generator is trained to fool the discriminator and produce realistic samples. Convergence is reached when there is an equilibrium between the generator and the discriminator.

A Wasserstein GAN uses the discriminator as an emulator for the Wasserstein distance. Arjovsky et al. [2017] provide theoretical guarantees and experimental constraints regarding the OT theory.

7.1.2.2 Deep learning

A number of works used differentiable distances between distributions as loss functions for a generative neural network. Notably, the MMD distance is differentiable as long as its kernel is differentiable. Dziugaite et al. [2015] propose MMD nets that use MMD with Gaussian kernel as loss function, since its good theoretical properties have been proven [Gretton et al., 2012]. Li et al. [2015] also use MMD but as part of an embedded feature extractor. Li et al. [2017] propose a MMD-GAN in which the discriminator learns the kernel for the MMD distance.

Arjovsky et al. [2017] conduct a study of different distances to be used with generative networks. They conclude that the Wasserstein distance is more regular than other integral probability metrics such as the Jensen-Shannon or the Kullback-Leibler (KL) divergence. They believe the MMD distance may have the same downside depending on the kernel with a saturation regime when distributions are too different. In these cases, gradient descent techniques are poorly efficient.

Finally, WGAN-GP [Gulrajani et al., 2017] improves the Wasserstein GAN of [Arjovsky et al., 2017] through experimental adjustments, notably the addition of a gradient penalty loss and a change of the optimizer. This model moves away from the OT theory that led to WGAN but achieves a more stable convergence.

Deep domain adaptation has been actually particularly developed in computer vision: Zhu et al. [2017] propose Cycle-GAN, an architecture that permits to perform domain translation in both directions thanks to two generators and two discriminators. Teng and Choromanska [2019] propose an invertible autoencoder to replace the generator so that a single network is trained for domain translation.

GAN in HEP

Finally, it should be mentioned that GAN have been increasingly used in HEP, started by de Oliveira et al. [2017] for jet shower simulations at LHC: for faster data simulation of cosmic ray-induced showers in Cherenkov detectors [Erdmann et al., 2018], particle showers in calorimeters [Erdmann et al., 2019], event simulation at LHC [Martínez et al., 2020], simulation of Cherenkov detectors at LHCb [Maevskiy et al., 2020] or shower simulation at ATLAS [Ghosh et al., 2020]. It is interesting to note

that Matchev and Shyamsundar [2020] warn against the use of GAN as a replacement for usual data generators since their accuracy cannot exceed the one of training data.

In particular, Alanazi et al. [2020] propose a feature-augmented GAN for event generation notably for CLAS12 and EIC. In their model, additional high-level features are provided to the discriminator.

7.2 Domain adaptation of the particles momenta

In this section, we develop a process to assess correctly the performances of the proposed transparent machine learning models on real data. We use mapping techniques of domain adaptation to map the labeled simulated data onto the real data distributions.

However, the source and target data used to learn the mapping must come from the same process: indeed, real data comprise additional background processes that are difficult or time-consuming to simulate.

In the following, we establish a training dataset that comprises a single control process. We make two hypotheses:

- First, that the posterior distribution is not altered by the mapping T between the source and target: $p(y|X) = p(y|T(X))$;
- Second, we assume there exists a single mapping T that applies for all processes. Namely, if we learn T on a specific process present in CLAS12 data, this T can be reliably applied on other processes such as DVCS or π^0 production, as long as they share the same domains for the input variables.

7.2.1 Constitution of a training dataset: control process

Actually, π^0 production events are easier to isolate in data compared notably to DVCS events. In particular, exclusive events, namely with measured electron, proton and two photons can be extracted with almost no remaining background. As long as the selection process is the same in simulation as in real data, two corresponding datasets are obtained. However, this selection must not be too tight so that the dataset is representative of the global distribution. The objective is then to learn a mapping T between the source dataset and the target dataset. The former consists in two-photons π^0 production events in simulation and the latter in two-photons π^0 production events in real data. This mapping is then to be applied on other processes.

The cuts used to select exclusive two-photons π^0 production events are the following:

- the missing energy $ep \to ep\gamma\gamma$ must be below 1.8 GeV in absolute value;
- the squared total missing mass $ep \to ep\gamma\gamma$ must be below 0.2 $\mathrm{GeV}^2/\mathrm{c}^4$ in absolute value;
- the invariant mass $\gamma\gamma$ must be between 0.09 $\mathrm{GeV}/\mathrm{c}^2$ and 0.18 $\mathrm{GeV}/\mathrm{c}^2$;
- the squared missing mass $ep \to e\gamma\gamma$ must be between 0 and 1.6 $\mathrm{GeV}^2/\mathrm{c}^4$;
- the total missing transverse momentum must be below 0.14 GeV/c.

Histograms of a few common low-level and high-level variables are displayed on Figure 7.5. They permit to apprehend the distance between the source and target distributions.

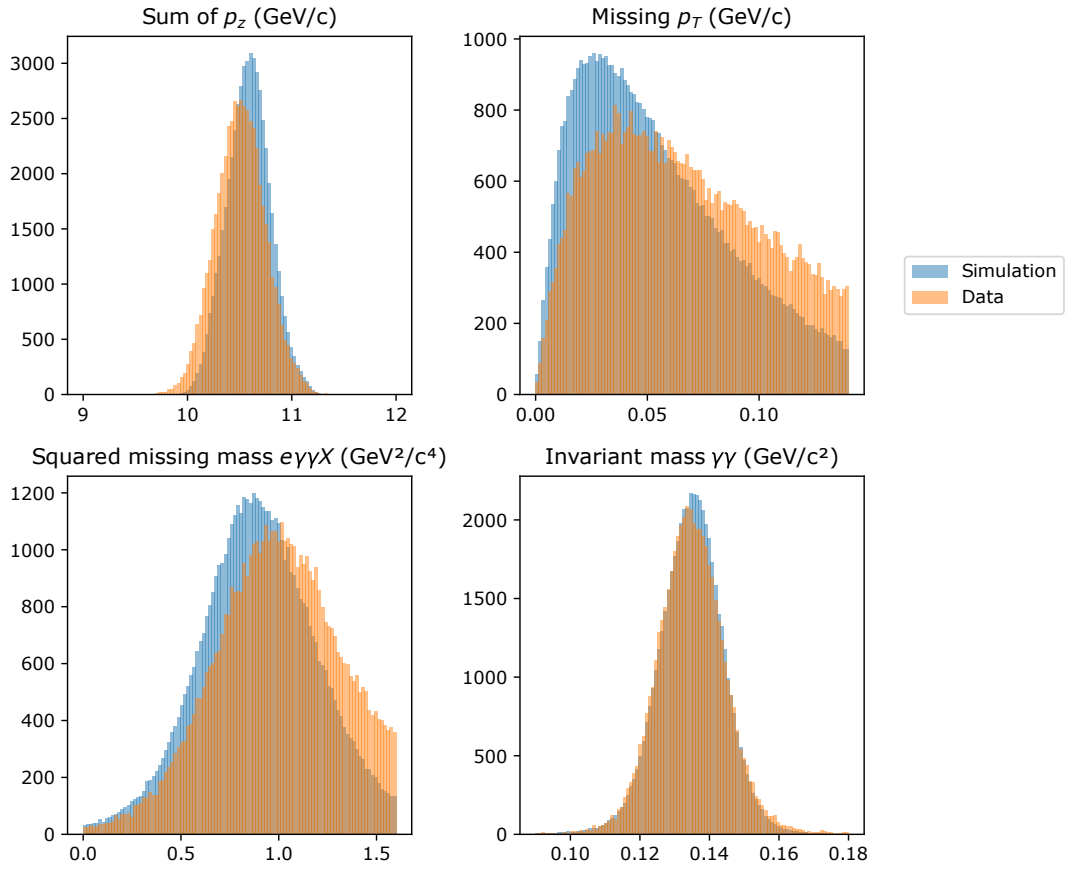


FIGURE 7.5: Histograms of simulated (source) and real (target) π^0 production events.

To control the proper functioning of the domain adaptation, we design a smeared dataset that will play the role of real data during the development phase. From a simulated dataset with cross-sections containing both DVCS and π^0 production events, we alter the distributions to imitate degraded detector resolutions and reconstruction biases. Noting p_{gen} the generated momentum of a given particle, p_{rec} its reconstructed momentum and p_{sm} the smeared momentum, we modify the momenta of the electron, proton and photons as follows:

- for charged particles in the forward detector (namely the electron and sometimes the proton): $p_{sm} = p_{gen} + 3.2\,(p_{rec} - p_{gen})$;
- for charged particles in the central detector (sometimes the proton): $p_{sm} = 0.9\,(p_{gen} + 2.5\,(p_{rec} - p_{gen}))$;
- the momenta of photons is drawn from a Gaussian distribution $\mathcal{N}(p_{rec},\, 0.05\,p_{rec})$

In addition, the direction of the momentum of all particles is also smeared by an angle drawn from a Gaussian distribution of standard deviation 1°: $\mathcal{N}\left(0, \frac{\pi}{180}\right)$. This smearing is performed in an arbitrary direction from p_{rec}. Histograms of the smeared data compared to the original simulated data are displayed on Figure 7.6.

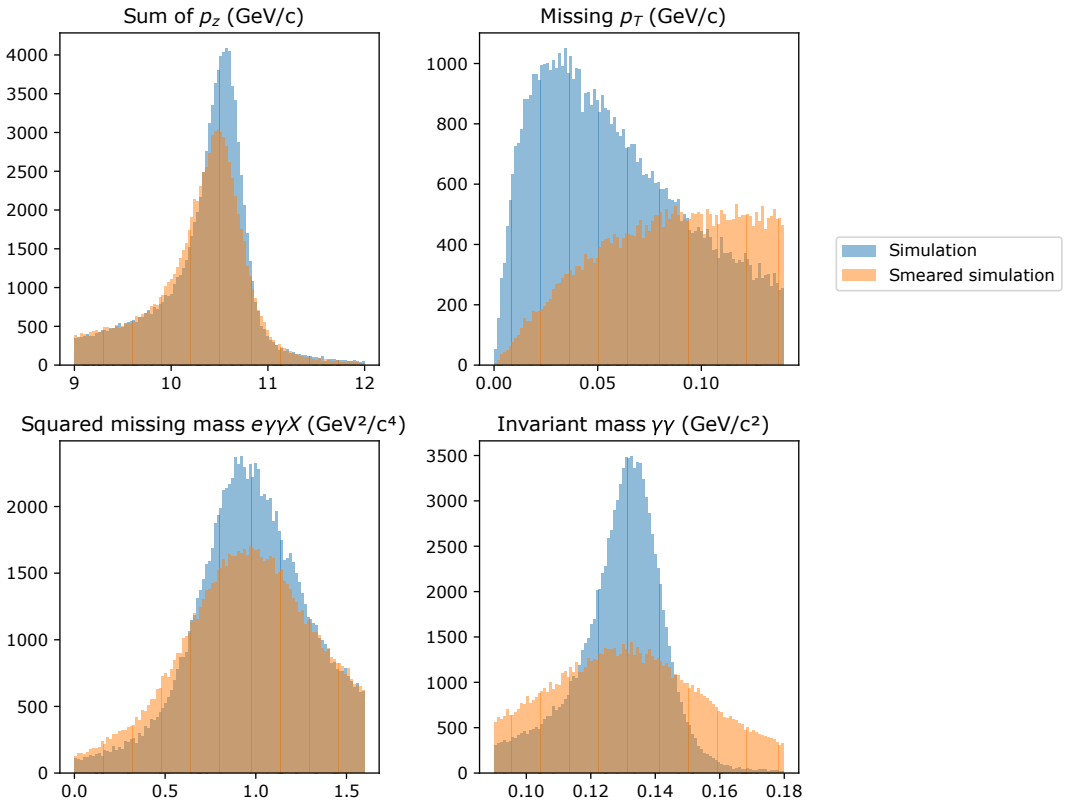


FIGURE 7.6: Histograms of simulated and smeared π^0 production events.

We then dispose of a source dataset and a target dataset, both containing DVCS and π^0 production events. However, the mapping T will be learnt exclusively on selected π^0 production events according to the list of cuts presented above. Applying a few classifiers on simulated and smeared data gives the ROC curves displayed on Figure 7.7. We use two models for this ROC comparison:

- a fuzzy C4.5 (Fibo version) with 25 high-level features automatically built in an embedded manner;
- a GAM of 16 terms with embedded feature construction and enforced bitonicity where the features are themselves bitonic (FCGAM_b_{max}).

Both models are initially trained on simulated flat data.

The objective of the domain adaptation is to transport the simulated data so that the classifiers applied to the transported data get the same performances as on smeared data, namely to reproduce the ROC curve of smeared data with the transported data. This would guarantee that the mapping T learnt on π^0 production events also extends to DVCS events. In addition, the performances of the models can be assessed on smeared data using the transported data. Therefore, the domain adaptation technique would be usable with real data as target. The models could be retrained or transferred on the transported data to improve their performance on real data.

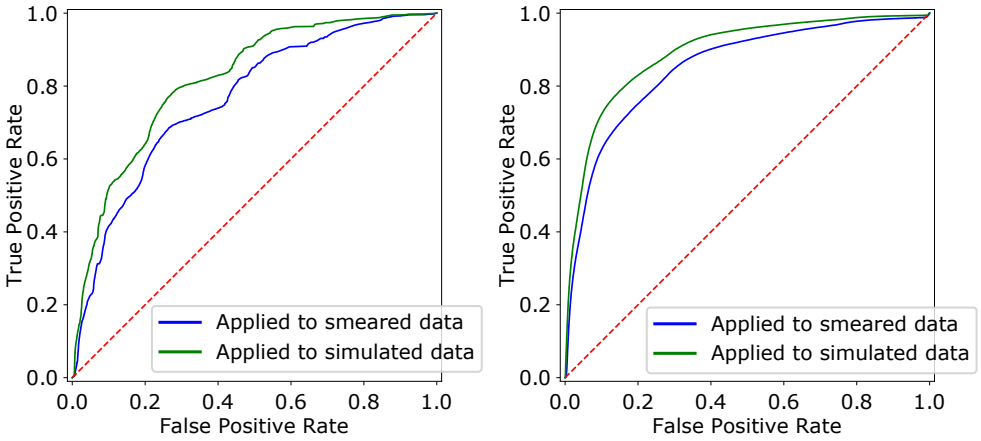


FIGURE 7.7: ROC curves of two transparent models trained on simulated flat data and applied to simulated cross-sections data and its smeared counterpart. Left: for a fuzzy C4.5 Fibo with 25 built features; right: for a FCGAM_b_{max}.

7.2.2 Proposed method for domain adaptation from simulation to CLAS12 data

From the review presented above, we retain a few domain adaptation techniques for CLAS12 domain adaptation: baseline methods such as renormalization and CORAL [Sun et al., 2015], regularized optimal transport [Cuturi, 2013], MMD-nets [Dziugaite et al., 2015] and WGAN-GP [Gulrajani et al., 2017]. This selection has been made considering that we need a mapping that conserves the original data structure and hence its interpretability. Revertible GAN are not considered here since we are looking for a mapping from simulated data to real data with degraded resolutions: the inverse path from real data to simulation is not possible.

We know that each type of particle can be transported independently. Indeed, the function mapping a simulated electron to its real counterpart does not depend on other particles produced at the same time. Therefore, the methodology to find a mapping for each particle is displayed on Figure 7.8: independent mappings are learnt for each type of particle (electron, proton, photon), but the evaluation (and training when possible) is common. Indeed, we follow the idea of Alanazi et al. [2020] to include additional high-level variables in the evaluation of the mappings to better constrain the training. The mappings only learn a transformation of the three-vector (p, θ, ϕ) of the particle. Then, the collective evaluation takes as input the transported three-vectors of all particles as well as additional high-level variables.

The chosen high-level variables are a mixture of frequent features built by the automatic feature construction and of common variables used by physicists:

- $\sum p_z^e + p_z^p + p_z^{\gamma_1}$ (noted $\sum p_z$ in the graphs);
- $\sin(\phi^e - \phi^{\gamma_1})$;
- $\cos(\phi^e - \phi^{\gamma_2})$;
- $\cos(\theta^e - \theta^{\gamma_1})$;
- missing energy $ep\gamma X$;
- squared missing mass $ep\gamma X$;
- photon cone angle;
- squared missing mass $e\gamma X$;
- missing energy $ep\gamma\gamma X$;
- squared missing mass $ep\gamma\gamma X$;
- π^0 cone angle;
- invariant mass $\gamma\gamma$.

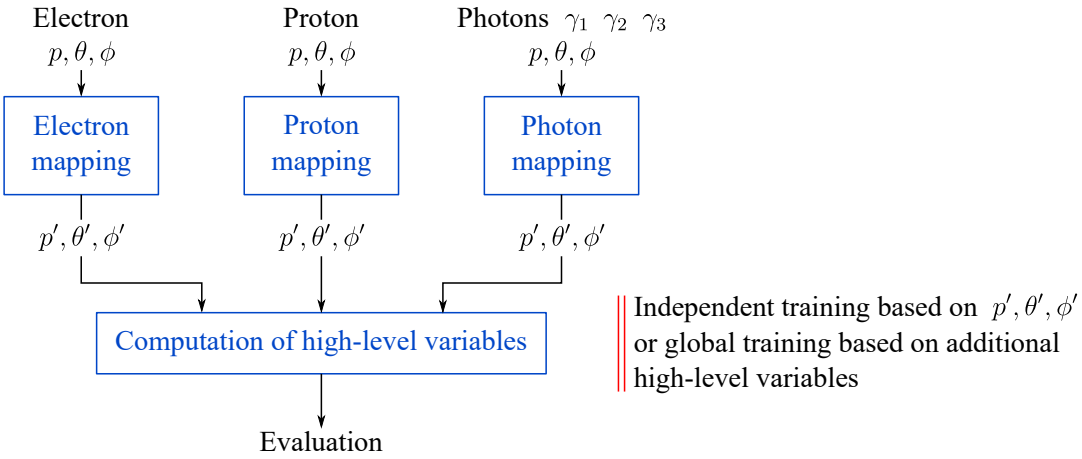


FIGURE 7.8: Methodology for CLAS12 domain adaptation.

From there, we decline the methods from the state of the art. The normalization is intrinsically independent for each variable. CORAL and optimal transport are applied on each particle separately. The training cannot take into account high-level variables for these three methods. Therefore, only the evaluation will consider them.

The details for our adaptation of MMD-nets and WGAN-GP are illustrated on Figure 7.9. The training of these networks takes into account the high-level variables as long as they are differentiable, which is the case.

As illustrated on Figure 7.9, we also try to simply learn a correction to the input data by including an additive loop (green on Figure 7.9): the generator networks should only produce a small correction to be added to the input data. With this idea, we include prior knowledge that the transported events should remain individually close to their source events.

Finally, we propose an alternative to these networks by adding random variables as input in addition to the three-vectors (p, θ, ϕ) (in blue on Figure 7.9). Indeed, part of the domain shift stems from overestimated detector resolutions, hence producing random Gaussian effects.

7.3 Experiments

7.3.1 Experimental settings

Normalization and CORAL do not have any hyperparameter since they simply use the statistics of the source and target data.

For the other methods, hyperparameters are determined based on the two-sample Kolmogorov-Smirnov (KS) distance[1]: we want classifiers to behave similarly on transported data and on target data. Therefore, we make use of the two same models applied to obtain the baselines of Figure 7.7: a fuzzy C4.5 Fibo with 25 built features and a FCGAM_b_{max}. We apply these models on the validation dataset, comprising exclusively π^0 production events. The output distributions of these models applied to the transported validation data and to the target validation data are compared

[1]The Kolmogorov-Smirnov distance is a non-parametric distance between the cumulative distributions functions of two one-dimensional probability distributions. The associated statistical test permits to assess whether two distributions are significantly different from each other.

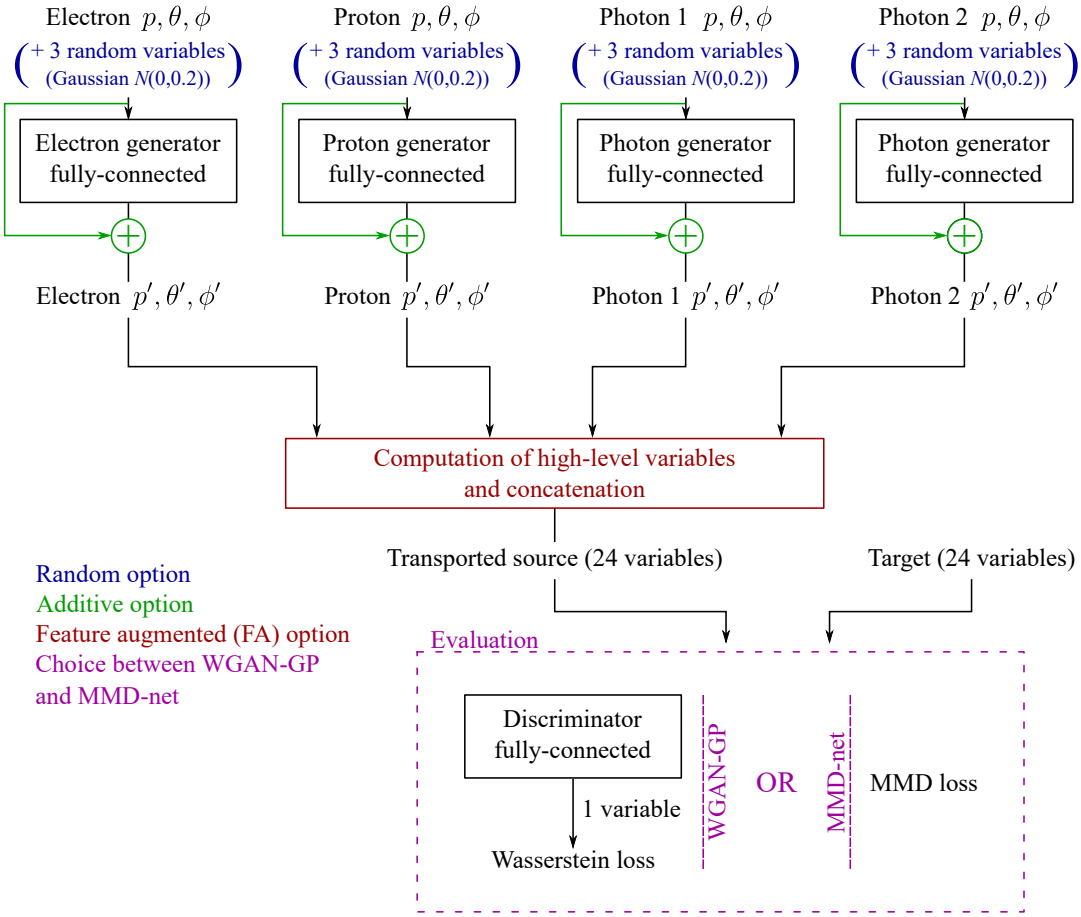


FIGURE 7.9: Adaptation of MMD-nets and WGAN-GP to CLAS12 domain adaptation.

with the KS distance. Returning to hyperparameter tuning, we choose the parameters minimizing the two KS distances (one for the fuzzy C4.5 Fibo and one for the FCGAM_b_{max}). Among two candidates in the Pareto optimum, we prefer the one for which the gains in KS distances are balanced between the fuzzy C4.5 Fibo and one for the FCGAM_b_{max}, namely we consider the ratio:

$$r = \frac{\mathrm{KS}_{\text{Fuzzy C4.5}}(\text{target,source}) - \mathrm{KS}_{\text{Fuzzy C4.5}}(\text{target,transported})}{\mathrm{KS}_{\text{FCGAM}}(\text{target,source}) - \mathrm{KS}_{\text{FCGAM}}(\text{target,transported})} \tag{7.5}$$

and choose a set of hyperparameters in the Pareto optimum so that r or $\frac{1}{r}$ is the closest to 1.

Using this principle, the regularization parameter for optimal transport is chosen by grid search over the range $[0.01, 10]$ with a multiplying factor of 3.33 between the successive trials. It varies as function of the training dataset and will be specified in the dedicated sections in the following.

For WGAN-GP, we start with the parameters used by Gulrajani et al. [2017] for their toy tabular dataset (experiments in their paper concern images), and perform a grid search in the neighborhood of these parameters to finetune the architecture and batch size. These specific hyperparameters vary in the following experiments and will be specified in the dedicated sections. All models comprise a certain number of dense

layers of 512 neurons each with leaky ReLU of slope 0.2 as activation function and dropout rate 0.1. The optimization is made via an Adam optimizer with learning rate 10^{-4}, $\beta_1 = 0.5$ and $\beta_2 = 0.9$ for both the generators and the discriminator. The training ratio is set to 5, namely there are 5 updates of the generators' weights for 1 update of the discriminator's weights. The discriminator loss is expressed as the mean of the product between the discriminator output and 1 for target samples, -1 for transported samples, plus a gradient penalty loss. The discriminator should learn to output a negative value for true target samples and a positive value for transported samples. Therefore, the discriminator is trained to maximize the distance between transported samples and target samples. On the contrary, the global generator (i.e. concatenation of the individual particle generators) loss is the mean of the discriminator outputs for the generated samples. In this way, the generator is trained to fool the discriminator (i.e. the discriminator should output negative values such as for real target samples). The discriminator disposes of the high-level variables computed from the transported and target samples.

We found that these parameters were well suited to MMD-nets as well, except that there is no discriminator. The loss of the generators is the MMD with Gaussian kernel between the target and the transported data comprising high-level variables.

For the random versions of MMD-nets and WGAN-GP, three random variables with Gaussian distribution $\mathcal{N}(0, 0.2)$ are added to the input of the generators. Indeed, three random variables should reflect the three-dimensional smearing applied to each source particle.

The training is performed on π^0 production events with their corresponding smeared events, among which 20% are used for validation. We found that 1200 epochs are sufficient to achieve convergence. A higher number of epochs sometimes leads to overfitting (the MMD starts increasing on the validation set).

The results presented in the following are evaluations of these methods on simulated test sets of around 180000 examples. These sets comprise DVCS and π^0 production events in equal amounts. To evaluate the quality of the transport, two quantities are considered for both considered models (the fuzzy C4.5 and the FCGAM). Firstly, the KS distance is computed between the output distributions of the classifier applied on the target data and on the transported test data. Secondly, the area of the absolute difference between the two ROC curves (the one when the model is applied to target data and the one when it is applied to transported data) permits to confirm the results. This area, noted AUD (for Area Under Difference), writes:

$$\mathrm{AUD} = \int_{\mathrm{fpr}=0}^{1} |\mathrm{ROC}_{transported} - \mathrm{ROC}_{target}| \, d\mathrm{fpr}. \tag{7.6}$$

These four evaluation measures (two KS distances and two AUD, one for each model) are evaluated on 5 folds of the test sets and presented with their mean and standard deviation.

To summarize, we consider successively:

- the validation set, comprising exclusively π^0 production events, used to determine the hyperparameters and choose the best model;
- the test set, comprising both DVCS and π^0 production events, on which the AUD metrics can be computed.

7.3.2 Experiments with smeared simulated data with flat distributions

Here, we consider a training set comprising π^0 production events simulated with a flat distribution over the CLAS12 phase space. The validation set comprises 20% of such events, and the test set both DVCS and π^0 production events simulated with a flat distribution.

We only present the results of the additive networks with feature augmentation, with or without random inputs. Additional experiments have been performed on normalization, CORAL, optimal transport and neural networks without feature augmentation or additive loop and can be found in Appendix F.

Hyperparameters have been tuned with the method detailed above:

- additive FA MMD-net: 5 layers, batch size 512;
- additive FA WGAN-GP: 5 layers for both the generator and discriminator, batch size 512;
- additive random FA MMD-net: 5 layers, batch size 512;
- additive random FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 512.

Table 7.1 presents the KS distances on the validation set, namely the set used to choose the hyperparameters. The WGAN-GP networks performs better than the MMD-nets whatever the variant (random or not). The best model based on these validation scores is the additive random FA WGAN-GP with a KS distance dropping to 0.007 for the fuzzy C4.5 model and to 0.019 for the FCGAM. It should be noted that the significance threshold for the KS distance with this number of samples is 0.008, making the output distributions from fuzzy C4.5 between the target and transported data not significantly different.

TABLE 7.1: Results on the validation set for flat distributions (π^0 production events only).

	$\mathrm{KS}_{\text{Fuzzy C4.5}}$	$\mathrm{KS}_{\text{FCGAM}}$	$\min\left(r, \frac{1}{r}\right)$
Baseline	0.019	0.031	
Additive FA MMD-net	0.019	0.029	0
Additive FA WGAN-GP	0.008	0.021	0.91
Additive random FA MMD-net	0.019	0.027	0
Additive random FA WGAN-GP	**0.007**	**0.019**	0.92

Table 7.2 presents the KS distances and AUD metrics when the learnt mapping is applied on the test set generated with flat distributions. This test set comprises additional π^0 production events and DVCS events as well. All models permit to improve the KS distances on the test set. Considering also the AUD, the WGAN-GP variants with additivity, feature augmentation and optional randomness remain the best models overall.

The ROC curves of the fuzzy C4.5 and the GAM applied to the source dataset, target dataset, and transported dataset using the additive random FA WGAN-GP are displayed on Figure 7.10. The output distributions of the fuzzy C4.5 Fibo and

TABLE 7.2: Results on the test set generated with flat distributions (DVCS and π^0 production events).

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}
Baseline	0.104	0.047
Additive FA MMD-net	0.062	0.024
Additive FA WGAN-GP	**0.016**	0.023
Additive random FA MMD-net	0.058	0.019
Additive random FA WGAN-GP	0.067	**0.015**
	$AUD_{\text{Fuzzy C4.5}}$	AUD_{FCGAM}
Baseline	0.070	0.043
Additive FA MMD-net	0.048	0.036
Additive FA WGAN-GP	0.009	0.012
Additive random FA MMD-net	0.045	0.035
Additive random FA WGAN-GP	**0.008**	**0.007**

the FCGAM are plotted on Figure 7.11. On this figure, we see that π^0 production events (classified closer to 0) are better reproduced than DVCS events (classified closer to 1). We can infer that domain adaptation is tuned for π^0 production events and therefore cannot be extrapolated in the DVCS region that easily. Distributions of a few high-level variables are displayed on Figure 7.12. These figures confirm that the target distributions are better reproduced than with the source distributions.

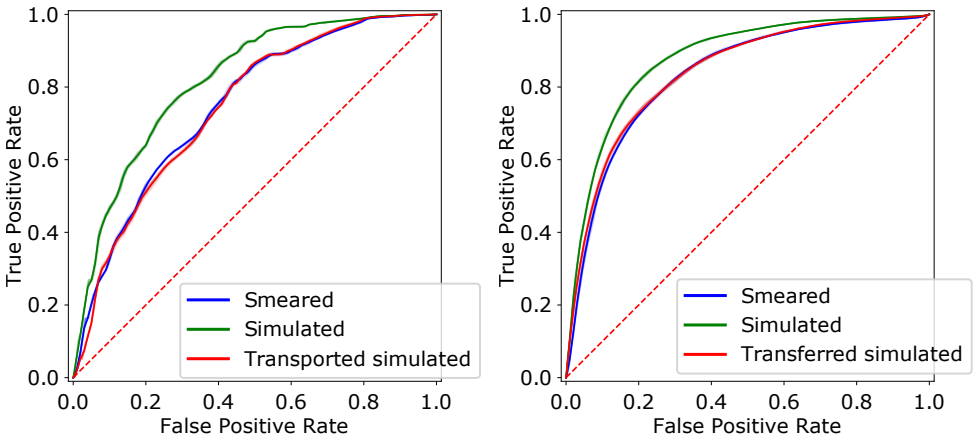


FIGURE 7.10: ROC curves of a fuzzy C4.5 (left) and a FCGAM (right) applied on the source dataset, target dataset and transported dataset (flat generation).

Arjovsky et al. [2017] mention that using the MMD distance as loss function is hazardous because of the shape of its gradient: indeed, the gradient quickly goes to 0 as soon as the two compared distributions are far enough from each other. Therefore, gradient descent is impractical on this distance. However, the considered source and target distributions are already quite close here. Therefore, the gradient is probably non-zero and gradient descent is possible, which explains the good results obtained by the MMD-net variants here.

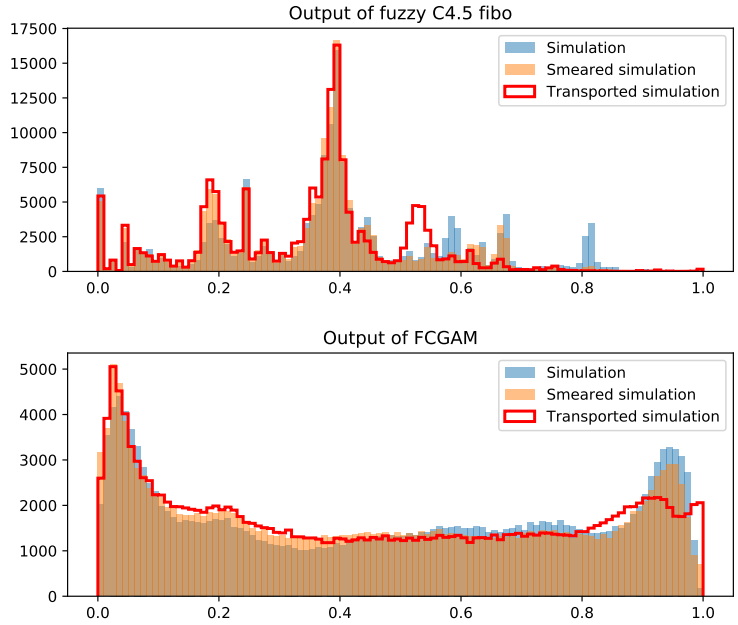


FIGURE 7.11: Distributions of fuzzy C4.5 (top) and FCGAM (bottom) outputs when applied on the source, target and transported datasets (flat generation).

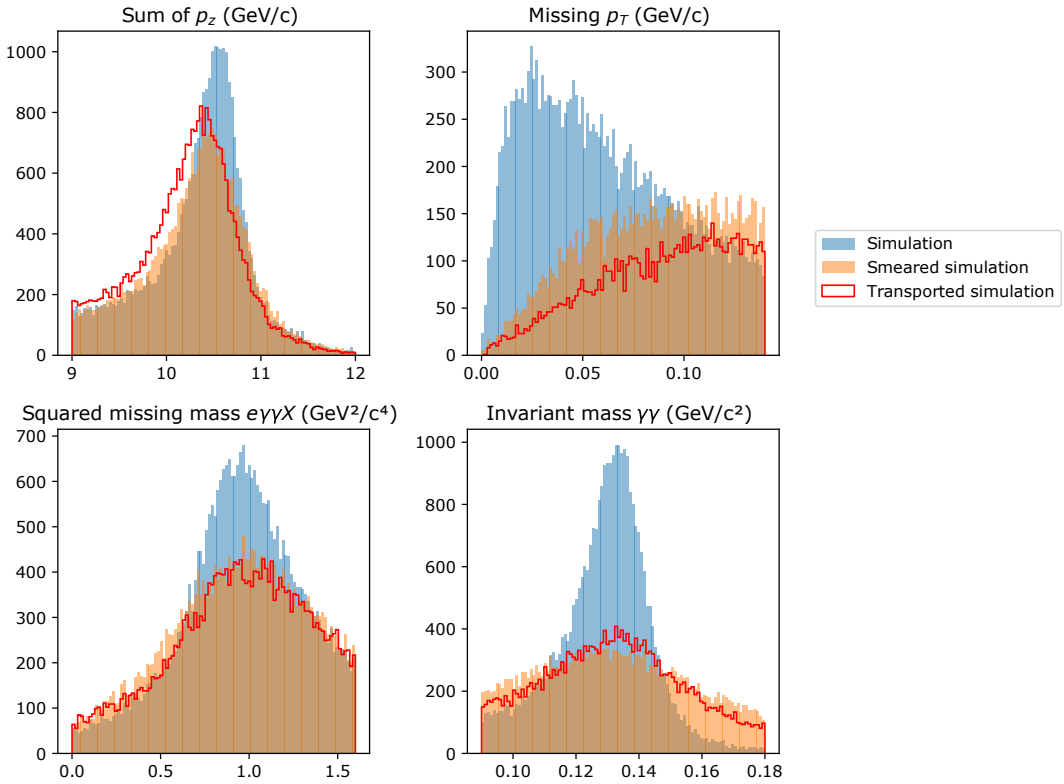


FIGURE 7.12: Histograms of simulated, smeared and transported π^0 production events (flat generation).

7.3.3 Experiments with smeared simulated data with cross-sections

Domain adaptation is now put into practice with a more realistic training dataset, consisting of exclusive π^0 production events simulated with cross-sections. Again, the validation set comprises 20% of such events. The test set is formed with DVCS and π^0 production events in equal amounts, simulated with cross-sections.

Hyperparameters with this training set are:

- additive FA MMD-net: 6 layers, batch size 256;
- additive FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 256;
- additive random FA MMD-net: 4 layers, batch size 256;
- additive random FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 256.

Table 7.3 presents the KS distances on the validation set. All models get an improvement of the KS distances for both the fuzzy C4.5 and the FCGAM. Here the MMD-nets obtain better results overall than the WGAN-GP networks. Overall, the best model considering a similar improvement on both the KS distances of fuzzy C4.5 and FCGAM is the additive FA MMD-net, closely with the additive random FA WGAN-GP.

Table 7.3: Results on the validation set for cross-sections distributions.

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}	$\min\left(r, \frac{1}{r}\right)$
Baseline	0.041	0.089	
Additive FA MMD-net	**0.017**	0.028	**0.39**
Additive FA WGAN-GP	0.020	0.029	0.35
Additive random FA MMD-net	0.022	**0.017**	0.26
Additive random FA WGAN-GP	0.018	0.029	0.38

Table 7.4 presents the KS distances and AUD metrics when the learnt mapping is applied on the test set generated with cross-sections. The generalized improvement of the KS distances observed on the validation set is not visible anymore here: the additive FA MMD-net, which was identified as the best performing model according to the validation scores degrades KS distance for FCGAM. The additive random FA WGAN-GP improves the AUD metrics but not the KS distances. Finally, the additive FA WGAN-GP degrades every metric while the random version of MMD-net is the only one to improve all of them. At first sight, the validation scores are here not correlated to the test scores and therefore to the generalization ability of the learnt mapping.

However, π^0 production events that were used for training here have been generated following the cross-sections: in practice, they are mostly produced in a region of the phase space quite different that the one covered by the DVCS events. π^0 production events generated with a flat distribution used in the previous experiments as training set provided a better coverage of the phase space. Therefore, the domain adaptation task was easier. Here, the extrapolation of the mapping to a part of the phase space

TABLE 7.4: Results on the test set generated with cross-sections.

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}
Baseline	0.090	0.047
Additive FA MMD-net	0.025	0.049
Additive FA WGAN-GP	0.138	0.141
Additive random FA MMD-net	**0.023**	**0.036**
Additive random FA WGAN-GP	0.096	0.058
	$AUD_{\text{Fuzzy C4.5}}$	AUD_{FCGAM}
Baseline	0.040	0.028
Additive FA MMD-net	0.011	**0.009**
Additive FA WGAN-GP	0.085	0.056
Additive random FA MMD-net	**0.007**	0.010
Additive random FA WGAN-GP	0.040	0.014

where training examples are rare is hazardous and may explain the disappointing efficiency of some mappings.

To further investigate, we observe the performances of the proposed domain adaptation techniques on the reduced phase space where π^0 production events are found in abundance. Figure 7.13 displays the distribution of π^0 production events in the (x_B, Q^2) plane for real data and simulated data, as well of the subspace that will be focused on in the following. This subspace is chosen to coincide with the kinematic bins that will be used for the analysis in chapter 9:

- $0.16 < x_B < 0.26$ and $Q^2 < 2.4\,\text{GeV}^2/\text{c}^4$;
- or $0.26 < x_B$ and $Q^2 < 3.25\,\text{GeV}^2/\text{c}^4$.

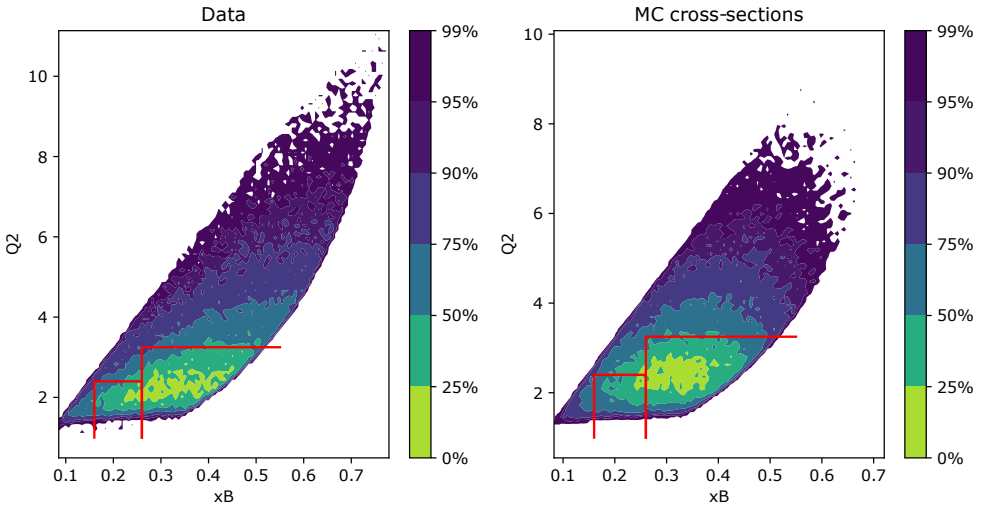


FIGURE 7.13: Distribution of exclusive π^0 production events in CLAS12 data on the left and in simulation on the right. In red are the boundaries of the two bins that are considered for further investigation.

We constitute a reduced test set with 50% π^0 production events and 50% DVCS events, each process being generated following the cross-sections in the reduced phase space.

The performances of domain adaptation on this subset are displayed in Table 7.5. Now, more methods achieve a satisfying performance based on the evaluated metrics. The additive FA WGAN-GP still degrades the metrics, although we could not find any plausible explanation. Here, the additive random FA MMD-net seems the most stable reliable model. The better agreement of the distributions can be observed on Figure 7.14, although there is still room for improvement especially regarding the invariant mass $\gamma\gamma$.

TABLE 7.5: Results on the reduced test set.

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}
Baseline	0.080	0.043
Additive FA MMD-net	0.024	0.044
Additive FA WGAN-GP	0.088	0.098
Additive random FA MMD-net	**0.015**	**0.014**
Additive random FA WGAN-GP	0.063	0.068
	$AUD_{\text{Fuzzy C4.5}}$	AUD_{FCGAM}
Baseline	0.038	0.035
Additive FA MMD-net	**0.012**	0.015
Additive FA WGAN-GP	0.062	0.044
Additive random FA MMD-net	0.013	0.016
Additive random FA WGAN-GP	0.030	**0.013**

Finally, Figure 7.15 displays the ROC curves of the fuzzy C4.5 and the FCGAM when applied to the simulated dataset, smeared dataset and transported dataset with the additive random FA WGAN-GP, reduced to the two bins of interest. The ROC is closely approached for the both models. Figure 7.16 shows that the output distributions of the fuzzy C4.5 and FCGAM are in better agreement after applying the learnt mapping.

7.3.4 Domain adaptation to real data

Finally, we perform domain adaptation for real data. As target dataset, we use selected exclusive π^0 production events from CLAS12 data. As source dataset, we use simulated π^0 production events generated with cross-sections that went through the same cuts performed on real data (listed in 7.2.1).

The hyperparameters are determined the usual way:

- additive FA MMD-net: 4 layers, batch size 512;
- additive FA WGAN-GP: 5 layers for both the generator and discriminator, batch size 256;
- additive random FA MMD-net: 5 layers, batch size 512;
- additive random FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 512.

The KS distances for the validation set are displayed in Table 7.6. The best identified model according to the KS distances is the additive FA WGAN-GP. However, based

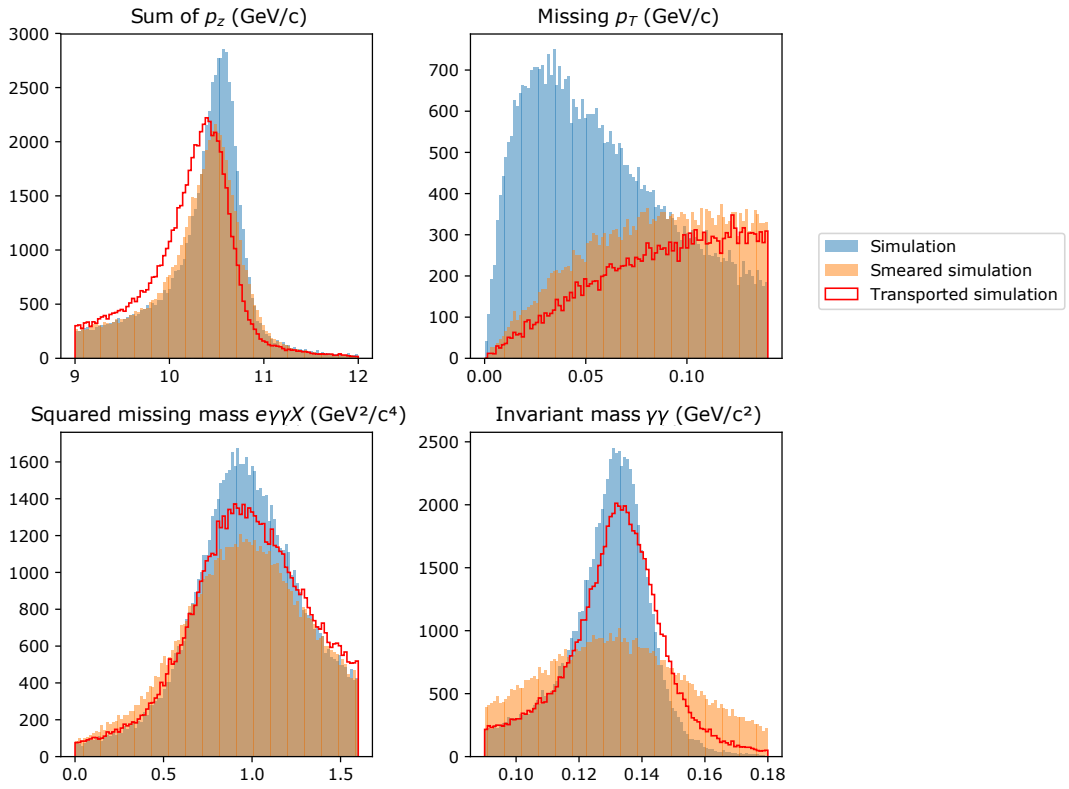


FIGURE 7.14: Histograms of simulated, smeared and transported π^0 production events with the additive random FA MMD-net (generation with cross-sections).

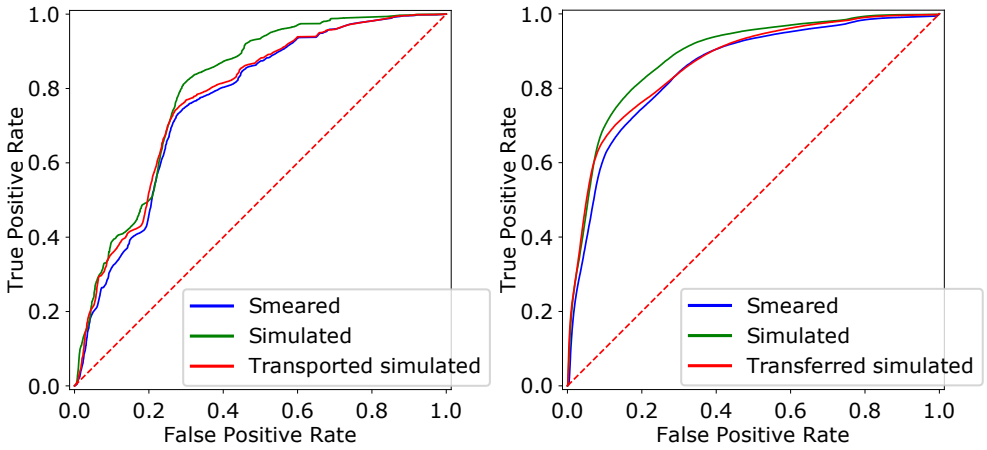


FIGURE 7.15: ROC curves of a fuzzy C4.5 (left) and a FCGAM (right) applied on the source dataset, target dataset and transported dataset (generation with cross-sections).

on the previous experiments, these validation scores must be taken with cautious and confirmed while looking at the high-level variables distributions. Figure 7.17 displays these distributions for the additive FA WGAN-GP. Since the additive random WGAN-GP has proven to be a more stable domain adaptation technique in the previous experiments, and since it also obtained good validation scores, the decision is also made based on visual improvement of the distributions: Figure 7.18 displays the distributions obtained by this means. The distributions are quite similar between

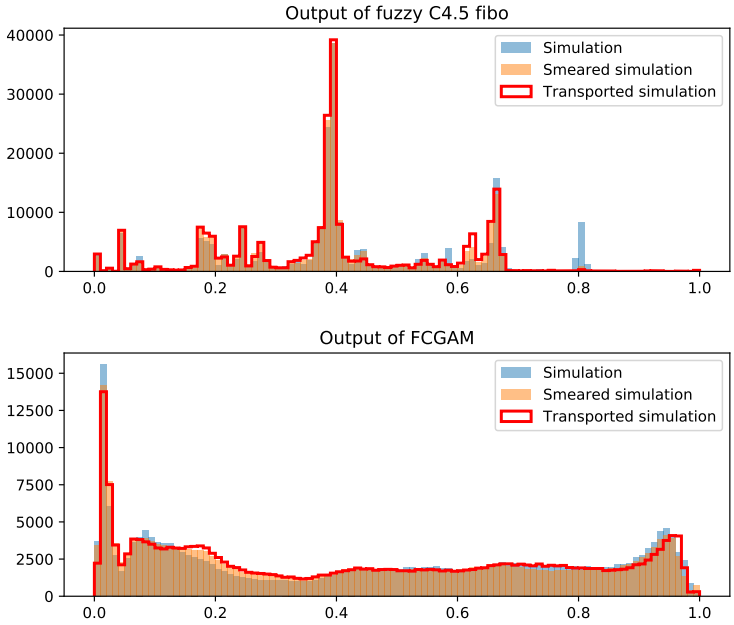


FIGURE 7.16: Distributions of fuzzy C4.5 (top) and FCGAM (bottom) outputs when applied on the source, target and transported datasets (generation with cross-sections).

the two techniques, but in better agreement for the additive random WGAN-GP notably looking at the squared missing mass $ep \to e\gamma`\gamma X$ or at the sum of the z momenta. Visibly, the distributions of missing transverse momentum and invariant mass $\gamma\gamma$ are both over-degraded. Therefore, it is probable that the ROC curves will be underestimated as well. Actually, the features used in the models should be studied in particular to evaluate the shift in their distributions, especially those in the first layers of the tree or in the first terms of the FCGAM.

TABLE 7.6: Results on the validation set for real data (π^0 production events only).

	$\mathrm{KS}_{\text{Fuzzy C4.5}}$	$\mathrm{KS}_{\text{FCGAM}}$	$\min\left(r, \frac{1}{r}\right)$
Baseline	0.054	0.068	
Additive FA MMD-net	**0.016**	0.052	0.42
Additive FA WGAN-GP	0.022	**0.035**	**0.97**
Additive random FA MMD-net	0.018	0.046	0.61
Additive random FA WGAN-GP	0.026	0.043	0.89

For the remaining experiments, we retain the additive random FA WGAN-GP as definitive domain adaptation technique.

7.3.5 Adapting transparent models to transported data

With the learnt mapping from source to target data, three options are open:

1. we can stay with the transparent models trained on flat simulated data, and assess their true performances on transported, realistic data;

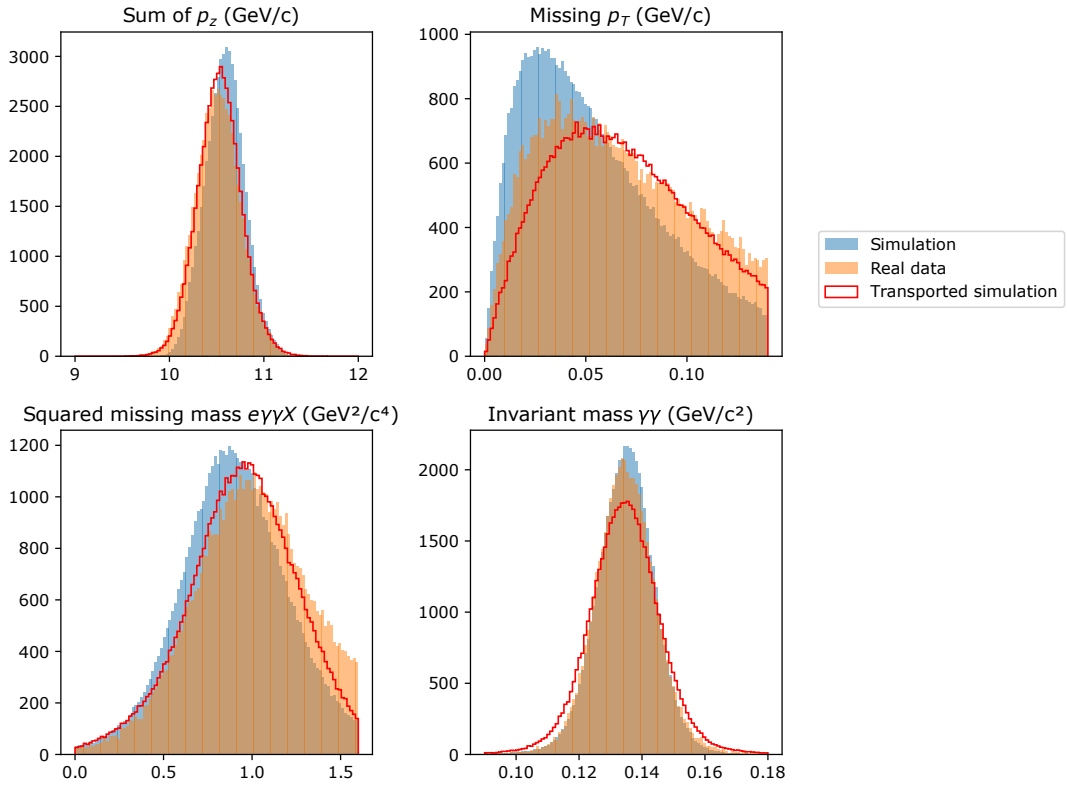


FIGURE 7.17: Histograms of simulated, real and transported π^0 production events with the additive WGAN-GP.

2. since simulated data is unlimited, a large transported dataset can be formed and transported models completely retrained on the transported dataset, including feature construction;

3. transparent models trained on simulated data can be partly transferred to the transported data, thus preserving the information learnt on an ideal simulation while avoiding biases from transported data.

We try these three alternatives for the two evaluation models used above: the fuzzy C4.5 and the GAM. While options 1 and 2 are straightforward, the methodology for option 3 depends on the considered classification model. The principle is to keep the built features and structure of the model, but refit the values associated to each feature. For C4.5, this means re-determining the split thresholds, width of fuzzification and potentially pruning some branches. For GAM this means re-inducing the shape functions with the same features and regularization parameters but different training data (here the transported dataset).

Model transfer or re-induction is performed on the transported flat dataset. Only the events belonging to the π^0-dominated subspace are retained since the generalization of domain adaptation cannot be guaranteed in the region of low π^0 density. Therefore, the learnt mapping is applied on the subset of the dataset generated with flat distributions that belongs to the two bins of the phase space where domain adaptation is reliable.

The ROC curves of the different options applied on the reduced flat dataset (transported with the mapping learnt with real data) are displayed on Figures 7.19 and 7.20

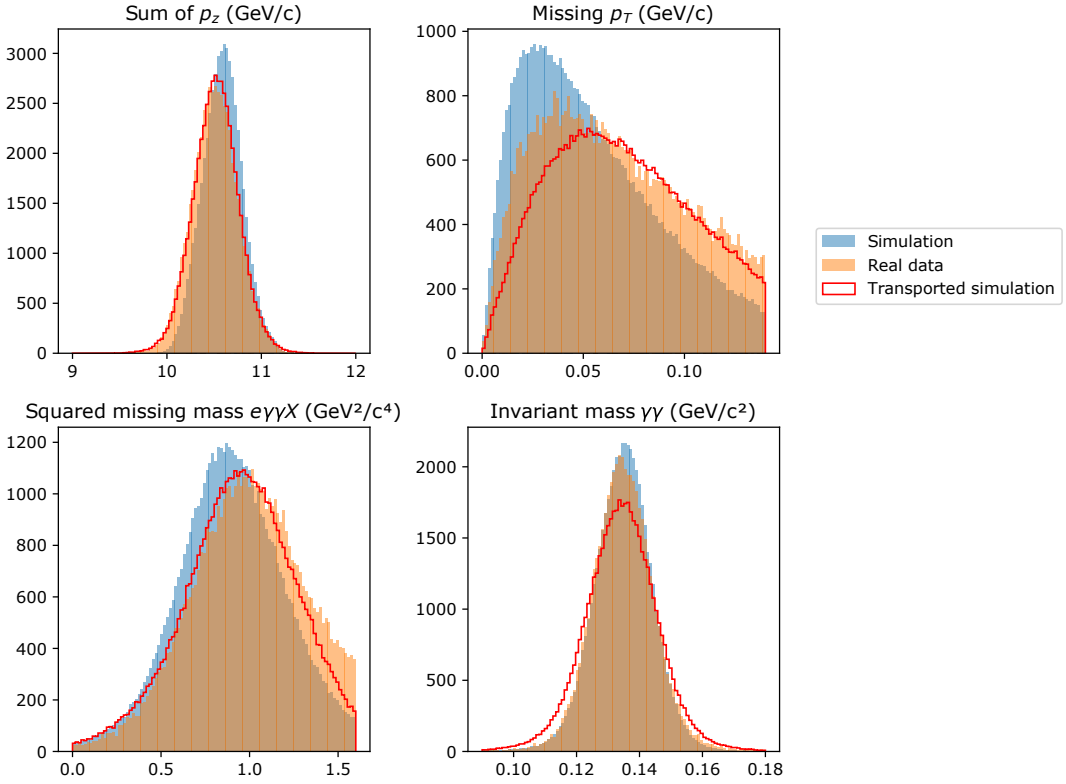


FIGURE 7.18: Histograms of simulated, real and transported π^0 production events with the additive random WGAN-GP.

for fuzzy C4.5 and GAM respectively.

The retrained models obtain the best classification performance. Indeed, they are specialized not only in the transported dataset that should reflect real distributions, but also in the reduced phase space. Regarding the transferred models, they do not necessarily perform better than the original versions. This is actually a good sign, since it means that the classification performances of the models are stable with respect to perturbations of the split thresholds. Moreover, depending on the selection threshold in the ROC curve, it may be possible to slightly increase the classification score.

To discuss the different options, option 1 (keeping the original model) is the one leading to the worse classification performance on real data. However, option 2 (retraining the model) is hazardous if the covariate shift is not the only shift that is corrected: indeed, π^0 cross-section models are not perfectly reliable on the considered phase space. Therefore, the learnt mapping probably corrects as well for the disagreement between the π^0 cross-section model and reality. Applying the learnt mapping on a dataset simulated with flat distributions will deform the phase space regarding this error on the cross-section model. Option 2 thus risks to produce a biased model. However, the gain in statistics may be enormous: up to 10% for the GAM at low false positive rates. Finally, retraining a model implies building new high-level features. The question is whether it is desirable or not to build features that rely on the quality of the target data. We could lose in interpretability since the features could be biased. Indeed, CLAS12 data currently contains biases due to imprecise calibration or empty regions in the phase space because of detector areas that are out of order. In addition, the resolutions of the detectors may vary with time, because of radiation damage for

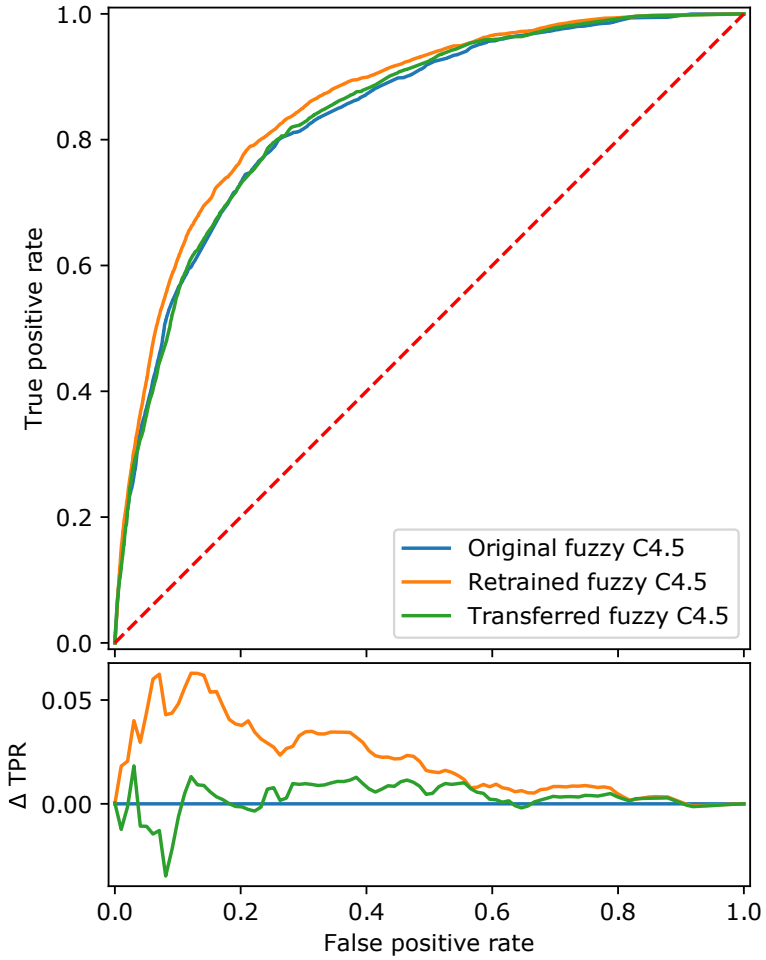


FIGURE 7.19: ROC curves of the original fuzzy C4.5 model, the transferred model to the transported data, and the retrained model on the transported data. Below are plotted the differences in true positive rates with the original model at fixed false positive rate.

instance. Option 3 (transfer the model) seems to be a good compromise to try to improve classification performance without biasing too much on the transported data.

7.4 Conclusion and perspectives

In this chapter, we covered the state of the art in domain adaptation to make up for the covariate shift between CLAS12 simulation and data. We constituted a training dataset consisting in exclusive π^0 production events, which are easily isolated in data. We tested several methods of the state of the art, concluding that the generative neural networks constitute the best option for domain adaptation. We proved the efficiency of the method on flat generated data, where the training dataset covers the entire phase space. This is especially true for the π^0 production sample. However, our assumption that the mapping derived from π^0 production events can be extrapolated to DVCS is not correct, despite the similarities between the two processes. Considering the cross-sections, the data used for training is located in majority in a specific region of the phase space. Therefore, domain adaptation does not generalize well elsewhere. However, the proposed methods remain a promising approach to correct for the distribution shifts between simulation and real CLAS12 data.

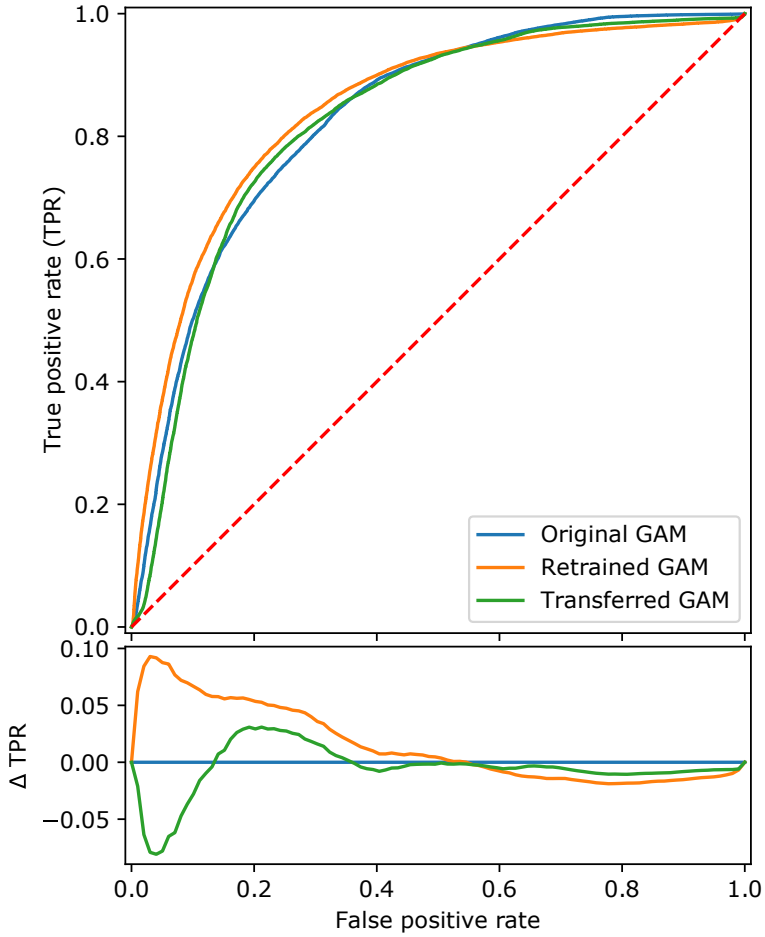


FIGURE 7.20: ROC curves of the original GAM model, the transferred model to the transported data, and the retrained model on the transported data. Below are plotted the differences in true positive rates with the original model at fixed false positive rate.

A first idea for further work is to imitate a large part of domain adaptation techniques building an invariant feature representation, but preserving interpretability. For instance, the proposed feature construction techniques of part II can be adapted to favor domain-invariant features with an additional term to the fitness computation. However, one needs to be sure of the quality of the data: indeed, the objective is to avoid biasing feature construction towards specificities of the CLAS12 actual setup. For instance, there currently exist dead areas in the detectors. To be generalizable, built features should not consider these details. We still think that encouraging feature construction to produce domain-invariant features is a good idea to facilitate or eliminate the need for domain mapping.

About the bias that can be created regarding the uncertainties of the π^0 production cross-section model, an idea would be to divide the phase space so that the difference in distributions between the source and target sample in terms of cross-section is negligible, leaving only the covariate shift. Therefore, a mapping between these two distributions would not be biased by the cross-sections. However, such a division is impractical since only 53232 events from real data are available, which is already limited to train the proposed models. With more events, such an approach would be feasible.

One main limitation to the proposed domain adaptation method is the phase space covered by the process used for training, namely exclusive π^0 production. Moreover, π^0 production events could even be easy to transport. Indeed, the calorimeter calibration in CLAS12 is performed with π^0 production events, which means it is the most optimal for low energy photons such as the ones that come from a π^0 decay. DVCS photons are usually much more energetic, therefore the calibration may not be optimal at their energies.

To get a training set that covers a larger phase space and improve the generalization ability of the proposed method, the following methodology could be applied: DVCS events actually cover the region of interest. However, obtaining a pure DVCS sample is complicated due to the π^0 contamination but not impossible. Indeed, using the method detailed in 9.2, we can subtract the π^0 contamination and retrieve distributions of almost pure DVCS events selected with strict cuts. The remaining work would be to adapt the proposed neural networks, taking negative weights for the π^0 contamination, thus embedding the subtraction in the domain adaptation. Domain adaptation can be performed on the concatenation of two sets: the set of exclusive π^0 production events already available, and the set comprising both strictly selected DVCS events (comprising π^0 contamination) and estimated contaminating π^0 events, the latter with negative weights. Therefore, the combination of both samples covers the entire phase space as well as the complete energy range of the photons. Hopefully, it will improve the capacities of the domain adaptation method.

Chapter 8

Interpretability evaluation by experimental physicists

8.1 Survey form 164
8.1.1 Evaluation of automatically built features 164
8.1.2 Evaluation of the interpretability of two given models . . . 165
8.1.3 Comparison among several models 166
8.2 Results and discussion 168
8.2.1 Profile of the respondents 168
8.2.2 Transparency requirements 170
8.2.3 Evaluation of the features 170
8.2.4 Evaluation of the models 175
8.2.5 Global comparison 181
8.3 Conclusion 184

As detailed in chapter 2 and more specifically in 2.1.3, the evaluation of interpretability is based on objective criteria but is also mostly done with respect to a specific application and to the target users. In this chapter, we follow the guidelines presented in 2.1.3 to rigorously evaluate the interpretability of the presented previously models.

Objective facts about the interpretability of the models have been progressively presented in part II, notably in 5.2.5 for automatically built features and respectively in 6.1.3 and 6.2.5 for decision trees and GAM with embedded feature construction.

To complete the objective remarks and because interpretability must be evaluated by the target users of the proposed models, we conduct an application-specific experiment with experimental physicists. According to the classification of Doshi-Velez and Kim [2017], our study is *application-grounded*: it concerns humans (here physicists) and real tasks (here event classification in CLAS12). However, Doshi-Velez and Kim [2017] state that large scale experiments are difficult to conduct since target users are experts of the field, therefore not numerous. Besides, human-produced explanations make a good baseline.

In this study, we designed a survey targeted to physicists to evaluate their perceived interpretability of the different components of the proposed models: the features automatically built by feature construction and the complete proposed transparent models. Finally, a comparison between transparent models, a neural network and a classical physicist analysis is conducted.

The survey, detailed in 8.1, has been sent to the CLAS12, COMPASS and Hall A international collaborations, as they are the most closely related to the DVCS analysis. Thus, all respondents were familiar with the physics context. The survey was divided into two parts: 31 physicists completed the first part and 24 the second part of the survey. The two parts were linked by a keyword provided by the respondents. Their responses are summarized and analyzed in 8.2.

8.1 Survey form

To design this survey, we followed the ideas of Mohseni et al. [2018] and Doshi-Velez and Kim [2017], presented in chapter 2, regarding the tasks that can be requested to specific types of users to evaluate at best interpretability of machine learning models.

We limit the survey so that it does not take more than one hour of the respondents' time. Since it demands concentration and thinking, the survey is split into two independent forms.

The survey is divided into three main parts:

- evaluation of automatically built features;
- evaluation of the interpretability of two given models;
- comparison among several models.

These three parts are detailed in the following. In addition, the profile of the respondents is studied through a series of questions regarding their status, age, field of expertise, etc. Their expectations regarding the use of machine learning in a physics analysis are also collected. Notably, the survey inquires about the degree of transparency necessary, according to them, for each subtask of a physics analysis. The complete survey can be found in Appendix G.

8.1.1 Evaluation of automatically built features

The goal here is to evaluate the interpretability of features built automatically by the feature construction algorithm described in chapters 5 and 6.

Considering the randomness of the feature construction process on the one hand, and the recommendation of Doshi-Velez and Kim [2017] to compare setups by pairs on the other hand, we establish the following evaluation protocol. First, we determine a list of pairwise matches that we want to make:

- probabilistic grammar-based GP (noted PGBGP) as feature construction method against the standard unconstrained GP-based technique;
- feature construction prior to model induction against embedded feature construction;
- automatic feature construction techniques (PGBGP prior or embedded) against regular variables used by expert physicists.

For each of these three matches, we randomly pick four features for each of the two involved categories, leading to a total of eight features. The respondent is then asked to give a mark between 1 and 5 to each feature, from poorly understandable to highly understandable, according to his perception of the physical meaning and of the relevance of the proposed feature. The respondent is encouraged to use the full range of

available marks, since the marks will be independent between each successive match. In total, the respondent will rate 24 features (8 for each of the three matches).

The marks obtained by each group are compared globally among all responses. If the difference is significant, then we conclude that one group is significantly more interpretable than the other.

8.1.2 Evaluation of the interpretability of two given models

Apart from the built features, we also want to evaluate the interpretability of the inducted models themselves. The chosen models are considered intrinsically transparent, but their reception by the physicists' community remains to be discussed.

This part of the survey is based on the framework proposed by Doshi-Velez and Kim [2017]. Notably, we take two ideas to verify the respondent's understanding of the model's functioning:

- the user is asked to simulate the model: a concrete instance is given to him and he should be able to determine how it is classified by the presented model;
- the user is asked to modify the model so that the output changes.

Since we are facing physicists who would maybe use these models someday, we also ask their opinion about the provided model, under the form of a mark between 1 (distrust or disagreement) and 5 (enthusiasm for using the model). The respondents are asked to think about the ease of validation, which is correlated with transparency. By validation, we imply retro-engineering of the decision process to understand the performances of the model in given regions of the phase space.

In addition to accuracy of understanding and subjective satisfaction, Lage et al. [2019] also suggest to measure the response time of the users. Regrettably, it is not a feature of Google Forms, used to organize the survey. We considered asking the respondent the hour at which he started reading the question and the hour at which he finished, but we finally renounced to avoid giving the impression of taking an exam.

We apply this evaluation protocol on two models: one parametric model and one vocabulary-based model. Evaluating more models would have led to a too long and time-consuming survey for the respondents. Moreover, most respondents were probably facing these models for the first time, even if we did not verify this assertion.

We choose a GAM with feature construction (algorithm developed in 6.2) for the parametric model, and a FURIA base for the vocabulary-based model. FURIA is preferred to a decision tree since it usually produces rules that are more compact. The instance used for model simulation is the same for the two models but is classified differently: the GAM classifies it as a background event while FURIA classifies it as a DVCS event.

Evaluating a GAM

For this evaluation, a GAM with feature construction and limited bitonicity enforcement (FCGAM_b_{max} as described in section 6.2) is trained on the flat CLAS12 simulation data with missing values (20000 training examples). The optimal number of terms for this model is 16, however we restrict the model to the first 5 terms in the survey to limit the time spent on it by the respondents. The 5-terms model is 3% less accurate than the full 16-terms model.

The model can be found in Appendix G. For the simulation question, the numerical values of a borderline example are provided. It suffices to remove a single term of the GAM to change the classification. Therefore, the respondents are first asked to classify the given example using the GAM model and then to remove one term of the GAM to change the output. Finally, the opinion of the respondents about the model's transparency is rated.

Evaluating a FURIA base

To have a similar complexity than the GAM model presented above, we consider a FURIA model with prior feature construction of five features. It is inducted on the same data, namely flat CLAS12 simulation data with missing values, using 4000 instances for the rule base induction.

The rule base can be found in Appendix G. The provided example, common to all proposed models, is classified as a DVCS example by the FURIA base. Two DVCS rules of the base are fired by this example against a single π^0 rule. Removing the DVCS rule with the highest confidence degree permits to reverse the classification of the base, since the remaining DVCS rule has a lower confidence value than the π^0 rule. Therefore, the respondents are asked the same questions than for the GAM: how does the rule base classify the example? Which rule should be removed to change the output? How transparent is the provided model according to the respondent?

8.1.3 Comparison among several models

Finally, the last part of the survey aims at assessing the physicists' overall preference on the proposed transparent models, compared to the standard physics analysis on the one hand and to a black-box model, here a neural network, on the other hand.

The physicist's cuts are provided by Guillaume Christiaens from the University of Glasgow who is working as a PhD student and performing the same analysis on the same dataset at CLAS12. He optimized these cuts to maximize the statistics (i.e. number of selected DVCS events) while minimizing the π^0 contamination on Monte

Carlo data. According to these cuts, an event is a DVCS event if:

$$\text{the missing mass } e\gamma \text{ is in } \left[0.1\,\mathrm{GeV/c^2}, 1.7\,\mathrm{GeV/c^2}\right] \tag{8.1}$$

$$\text{and angle } \left(\overrightarrow{p^{\gamma_1}}, \overrightarrow{e^{in}} - \overrightarrow{p^e} - \overrightarrow{p^p}\right) \leq 0.6° \text{ (photon cone angle)} \tag{8.2}$$

$$\text{and } \sqrt{(p_x^e + p_x^p + p_x^{\gamma_1})^2 + \left(p_y^e + p_y^p + p_y^{\gamma_1}\right)^2} \leq 0.12\,\mathrm{GeV/c} \tag{8.3}$$

$$\text{and the squared missing mass } ep\gamma \text{ is in } \left[-0.04\,\mathrm{GeV^2/c^4}, 0.04\,\mathrm{GeV^2/c^4}\right] \tag{8.4}$$

$$\text{and the missing energy } ep\gamma \text{ is in } [-0.5\,\mathrm{GeV}, 1.2\,\mathrm{GeV}] \tag{8.5}$$

$$\text{and angle } \left(\overrightarrow{p^e}, \overrightarrow{p^{\gamma_1}}\right) > 10°. \tag{8.6}$$

In addition, Marouen Baalouch worked as a post-doctoral researcher at CEA, LIST notably on black-box models to perform event selection. We use one of his models and hyperparameters, namely a two-layer fully-connected network with 20 and 30 hidden neurons respectively, with Adam optimizer with learning rate 0.05 and momentum 0.9. The model is trained for 60 epochs on 1.7 million events from flat CLAS12 simulation, with a batch size of 10000. It is represented in the survey along with a post-hoc explanation using SHAP [Lundberg and Lee, 2017] to quantify sensitivity to the different input features. It can be found in Appendix G. To ensure the respondents get a little familiar with the neural network, we ask the same questions than with the GAM and FURIA: we provide the same example and ask how it is classified by the neural network, given the SHAP values, and the opinion of the respondents about the model's ease of validation. Finding a modification of the model to change the output is intractable. Therefore, we do not include this question.

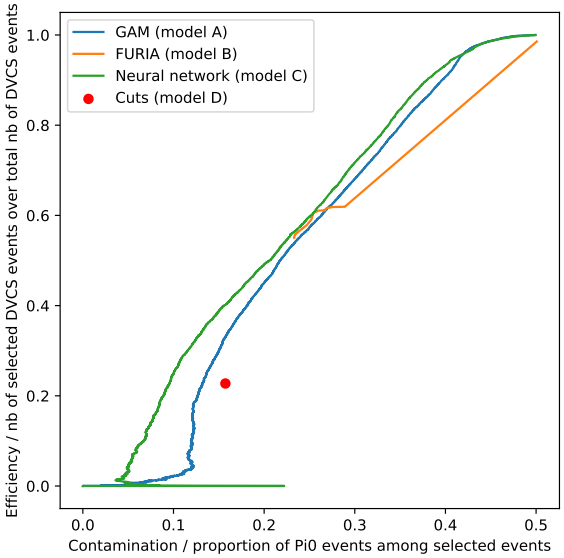


FIGURE 8.1: Selection efficiency as a function of π^0 contamination, for the four models presented in the survey.

After examining these four candidate models, the users are provided their performances on a test dataset of 100000 examples, under the form of a efficiency against contamination curve. In y-axis is the true positive rate, i.e. number of selected DVCS

events over the total number of DVCS events, and in x-axis is the π^0 contamination rate, i.e. the number of selected π^0 events over the number of selected events. This curve is displayed on Figure 8.1. The neural network obtains the best performance, followed by the GAM and the FURIA base. The cuts (red dot) get the worst performance with 23% efficiency and 16% contamination. However, they represent the approach physicists are the most used to.

It is specified in the survey that FURIA has a limited flexibility in the true positive/-false positive tradeoff. In addition, the survey mentions the π^0 subtraction step, which here is said to be able to remove the contamination provided it does not exceed 30%.

Then, the respondents are asked to rank the four models according to their subjective preference: which one would they use in practice for a physics analysis? How do they make their tradeoff between performance and interpretability?

8.2 Results and discussion

In practice, the survey has been split into two forms. The first one includes:

- the question about the respondent's expectations regarding interpretability of the used method for each step of the data analysis;
- two blocks of feature evaluation, i.e. PGBGP versus unconstrained GP, and prior versus embedded feature construction;
- GAM evaluation.

The second form comprises:

- a block of feature evaluation with automatically built features versus features designed by an expert physicist;
- FURIA evaluation;
- the global comparison between the GAM, the FURIA rules, a neural network and the classical analysis method;
- questions about the profile of the respondent.

A keyword has been asked to each respondent, to fill in at the beginning of each form, in order to associate the two parts.

The survey has first been sent to 6 beta-testers, whose feedback permitted to improve the clarity of the questions and explanations, notably the questions on the perceived transparency of the models and the global comparison. Thanks to them, we also noticed that the neural network needed its own questions, similarly to GAM and FURIA, so that the respondents better apprehend this model.

8.2.1 Profile of the respondents

31 physicists answered to the first form and 24 to the second. Since the profiling questions were located at the end of the second part, we know the characteristics of 77% of the respondents. The distributions of age and professional status properly cover all categories of the researchers population (see Figures 8.2 and 8.3). The two persons that answered "Other" to the status were respectively a data scientist and a retired staff scientist.

However, it was predictable that people who would answer the survey would be a priori interested by machine learning and not completely opposed to its usage. Indeed, 87.5% of the respondents declared being at least curious of machine learning (Figure 8.4) and nobody stated that they do not expect machine learning to change dramatically the reach of physics experiments (Figure 8.5). Actually, 50% of the respondents believe that using machine learning for the full reconstruction-analysis chain will significantly improve the physics output. These first questions demonstrate the largely positive attitude of the respondents towards machine learning.

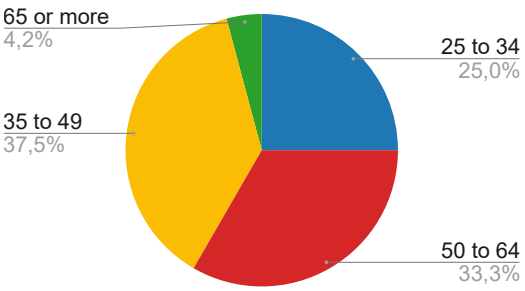


FIGURE 8.2: Age of the respondents.

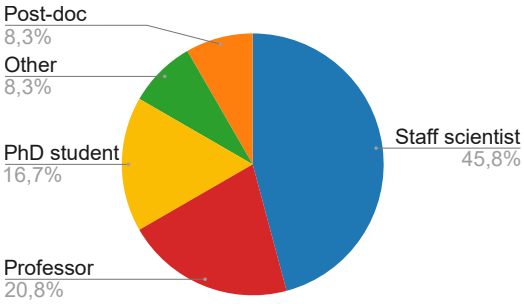


FIGURE 8.3: Professional status of the respondents.

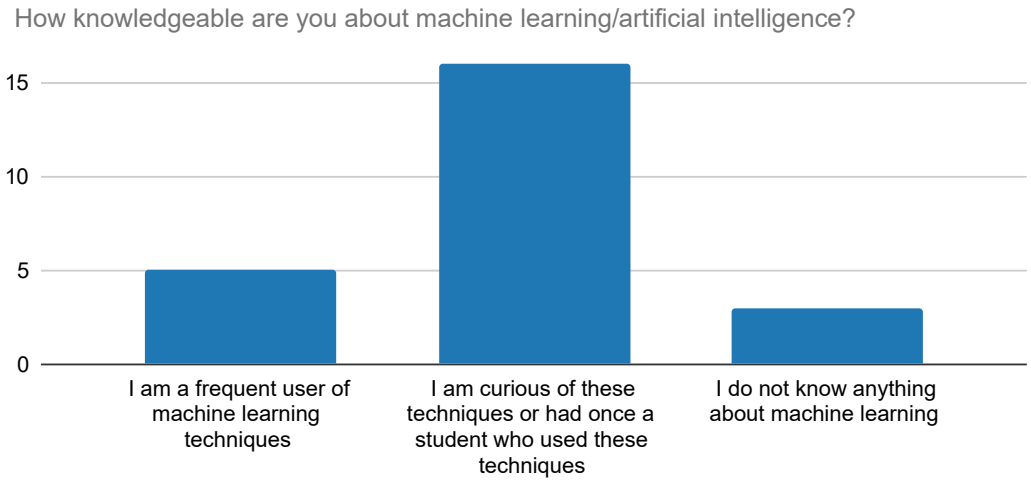


FIGURE 8.4: Respondents' knowledge about machine learning.

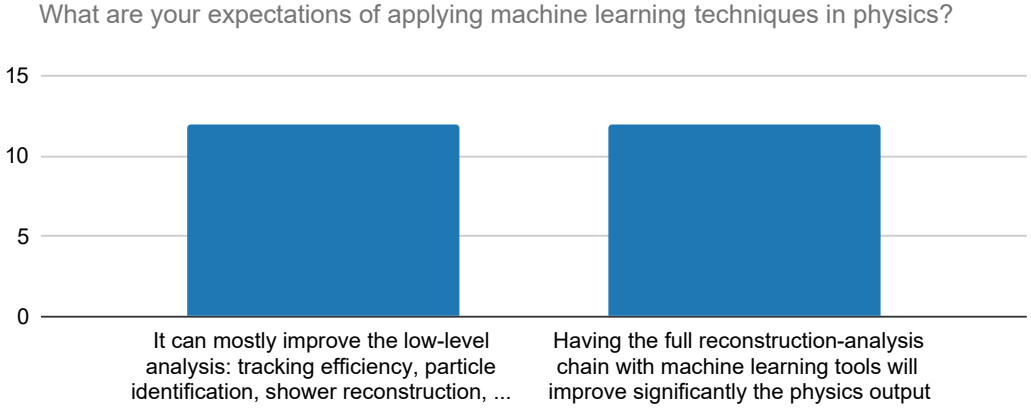


FIGURE 8.5: Respondents' expectations about machine learning.

8.2.2 Transparency requirements

The first question of the survey is about the respondents' requirements for transparency in different tasks of a physics analysis: tracking, particle identification, event selection and data smearing. Three options were offered to the respondents:

- no need for any explanation as long as the method is working well;
- the method should be sufficiently transparent to be validated;
- complete understanding of the method is far more important than performance.

The results are displayed on Figure 8.6. Globally, the participants seem to prefer the compromise between performance and transparency for all tasks. However, some discrepancies appear between the tasks: tracking is the task for which respondents prefer the most performance to transparency. On the opposite, transparency is required the most for particle identification and event selection. The last item, "smearing simulation to imitate data", has been probed to evaluate whether domain adaptation from simulation to real data should be interpretable from a physicist's point of view. As a result, the request for transparency is reduced compared to event selection or particle identification, but stronger than for tracking. Indeed, it has been shown in chapter 7 that relying on the generalization power of an opaque technique for domain adaptation is hazardous.

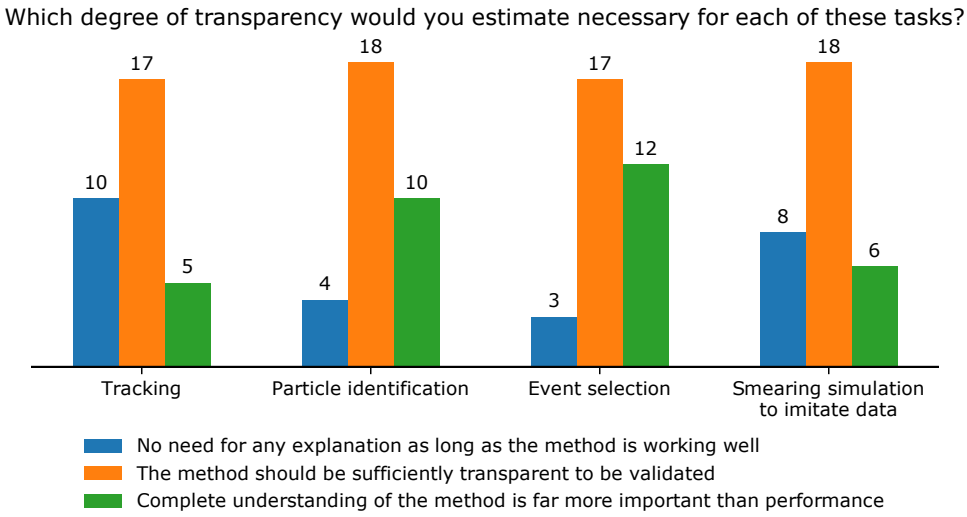


FIGURE 8.6: Transparency requirements of the respondents for different tasks of a physics analysis.

8.2.3 Evaluation of the features

In three groups of eight features, categories of features have been evaluated against each other using a scale of 5 steps (from 1: poorly understandable; to 5: highly understandable). We detail the results for each group first and make global comments afterwards.

8.2.3.1 Constrained against unconstrained feature construction

Figure 8.7 displays the results of the block of features containing four features built with a constrained GP algorithm and four features built with the unconstrained GP algorithm. It is clearly visible that constrained features have been better rated than the unconstrained ones: the average score for the unconstrained feature construction is 1.69, against 3.19 for constrained feature construction. One exception subsists though: the last feature has been built with constrained GP but is badly noted. The feature is:

$$\left(p_T^p + \left\|\overrightarrow{p^{\gamma_2}}\right\|\right) \tan\left(\text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_1}}\right)\right). \tag{8.7}$$

Although it respects the physical units, we can guess that the sum of a transverse momentum with a norm is complex to apprehend and to understand. However, no one made a remark in the comment section.

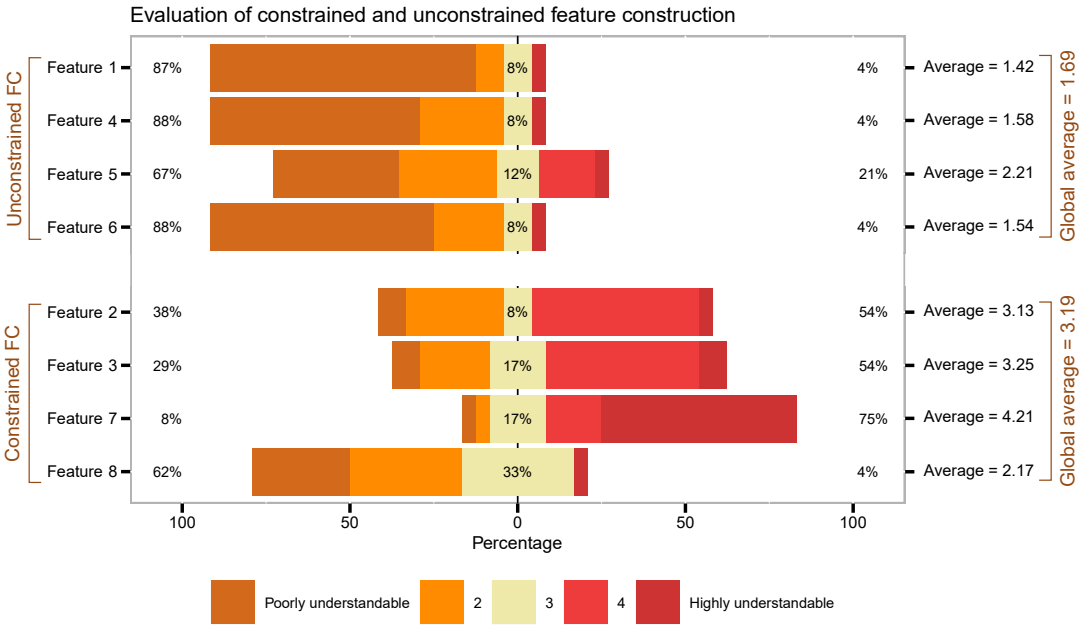


FIGURE 8.7: Results of the survey on constrained against unconstrained feature construction (FC). The distributions of the responses are displayed for each feature, along with the mean over all respondents. The average score for each category is printed on the right.

However, several respondents said they were puzzled by the proposed features. Here are a few examples among the seven comments for this question:

- "What is the meaning of a cosine of a momentum?"
- "Trigonometric functions acting on non-pure numbers are meaningless."
- "Not familiar enough to judge the understanding of the variables."
- "I think I am missing something – all of these seem really obscure, except the simple momentum balance. Also, cosine of a momentum? That seems odd..."

8.2.3.2 Prior against embedded feature construction

Figure 8.8 displays the results of the block of features opposing prior and embedded feature construction. No clear tendency emerges from the visualization of the results: the average score for embedded feature construction is 2.86, closely ahead of prior feature construction with 2.77. Performing feature construction in an embedded way does not seem to impair interpretability, while it is computationally more efficient than prior feature construction. However, our intuition is that four features are not enough to be representative of embedded feature construction: this category covers many algorithms (crisp and fuzzy decision trees, FURIA, GAM) and features built at different levels of specificities. Indeed, a feature built at the root of a decision tree has a high probability to be similar to a feature built with prior feature construction since it uses all available data. However, a feature built in a deeper node of the tree will be specific to the data subset that reached this node. Intuitively, the more specific the features, the less understandable they are.

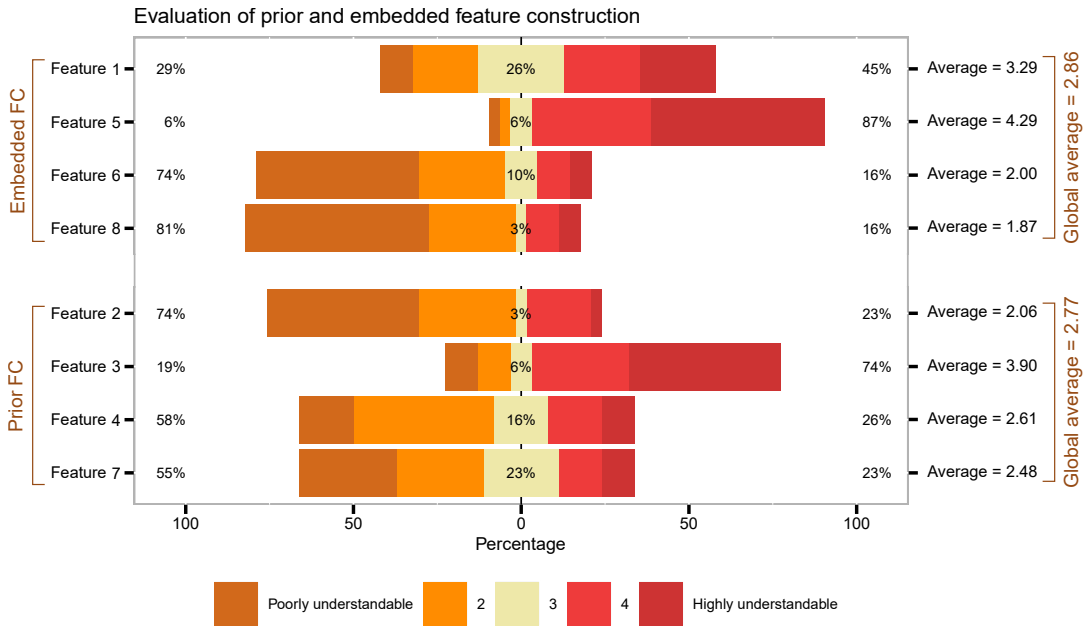


FIGURE 8.8: Results of the survey on prior against embedded feature construction (FC). The distributions of the responses are displayed for each feature, along with the mean over all respondents. The average score for each category is printed on the right.

One of the proposed features raised comments:

$$-\theta^p + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, 2\overrightarrow{p^e}\right) + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^p}\right). \tag{8.8}$$

The two comments about this feature are:

- "I cannot quite understand why the first variable's middle term is angle($\overrightarrow{p^{\gamma_2}}$, $2\overrightarrow{p^e}$). Is the coefficient of $\overrightarrow{p^e}$, '2', necessary?"
- "in variable 6, when calculating angle why instead of $\overrightarrow{p^e}$, the $2\overrightarrow{p^e}$ is used? the result should be the same right?"

Indeed, these comments underline the fact that the proposed feature construction algorithm does not look for simple features, it just optimizes their discriminative

power. When facing two features of equal fitness, the algorithm simply picks the first one in its list. This is obviously an area of improvement.

8.2.3.3 Automatic feature construction against regular physicists' variables

Figure 8.9 presents the results of the last comparison, namely automatic feature construction against regular variables used by physicists. Again, there is no significant difference between the two categories, but physicists' regular variables are still slightly ahead of automatically built features, with an average score of respectively 3.74 and 3.51. However, the score obtained by the two last physicists' regular variables (features 7 and 8 on Figure 8.9) must be discussed. Indeed, they are very common variables used by physicists for exclusivity cuts, but were written in the survey as mathematical formulas that might have not been recognized:

$$\text{missing mass } ep \to e\gamma X : \sqrt{\left(-\left\|\overrightarrow{p^e}\right\| - \left\|\overrightarrow{p^{\gamma_1}}\right\| + M_p + e_z^{in}\right)^2 - \left\|-\overrightarrow{p^e} - \overrightarrow{p^{\gamma_1}} + \overrightarrow{e^{in}}\right\|^2}, \tag{8.9}$$

$$\text{missing energy } ep \to ep\gamma X : e_z^{in} + M_p - \left\|\overrightarrow{p^e}\right\| - \sqrt{\left\|\overrightarrow{p^p}\right\|^2 + M_p^2} - \left\|\overrightarrow{p^{\gamma_1}}\right\|. \tag{8.10}$$

This last feature has indeed received contradictory scores, with 23% of respondents saying it is poorly understandable and 32% that it is highly understandable. With the textual formulations such as "missing mass" and "missing energy", probably these features would have received a higher score. However, comparing all features on equal terms indicates that features with complex mathematical formulations are considered less understandable.

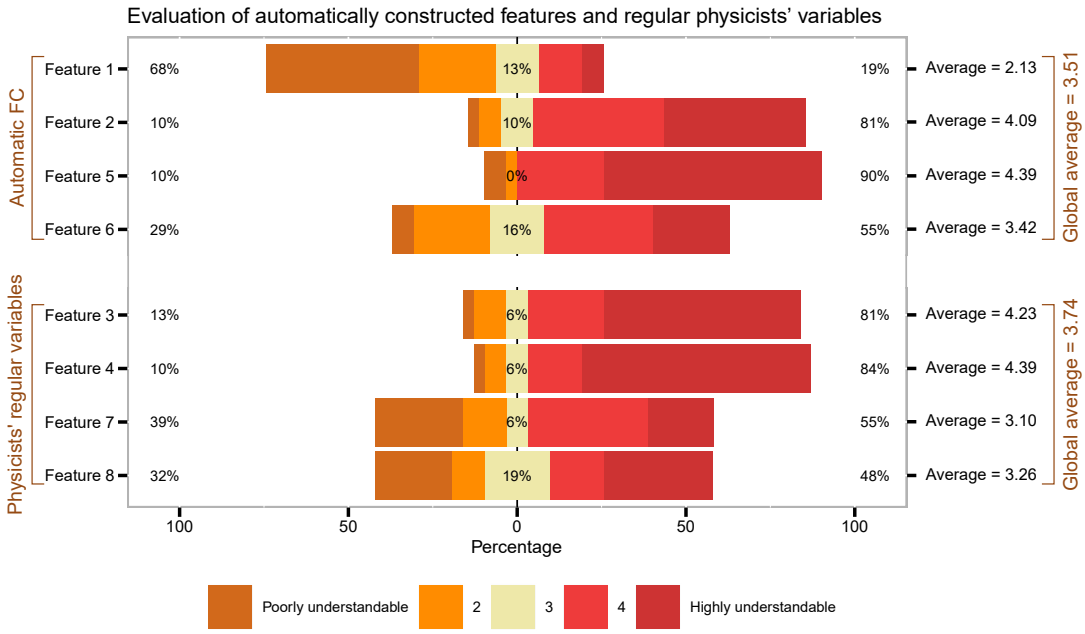


FIGURE 8.9: Results of the survey on automatic feature construction (FC) against regular variables used by physicists. The distributions of the responses are displayed for each feature, along with the mean over all respondents. The average score for each category is printed on the right.

Globally, there is no significant difference between the understandability of automatically built features and physicists' regular variables. Therefore, our proposed constrained feature construction algorithm permits to automate the feature engineering step without significant loss in interpretability. Moreover, the automatically built features were demonstrated more discriminant than physicists' regular features.

8.2.3.4 Global comments about the features

Some comments have been made that are not specific to a particular group of features. In particular, it has been noted through comments and also oral discussions with a few participants that some of them needed a proof of the discriminative power of the variables, more than a physical intuition of what they are.

- "The expressions are clear, the physical meaning is not clear sometimes, but I can understand the idea that those variables might have a good separation power based on whether the event is signal or background!"
- "Maybe with oral explanations I could understand more variables."
- "The understandability seems almost binary to me: either I understand what the variable is, and then it is absolutely clear, or I kind of see what the expression is doing but I do not see exactly why."

Orally, a few participants said that they would have liked being given the histograms of the proposed features to evaluate their discriminative power.

One participant went further in this idea:

> "It is highly unclear how the reader's vector algebra proficiency is relevant at all in how the machine learning algorithm is supposed to work. I always thought that the promise of machine learning is to let the computer find out features and combinations thereof that might escape a human. The fact that a variable makes or does not make sense or has an immediate physical interpretation (again, for humans) should not be a reason for preventing a machine learning algorithm from using it. That said this reviewer is highly skeptical of variables that combine several/many individually measured features, especially as there could/are correlations between these that might, or might not be fully taken into account and/or might be underestimated."

This last comment shows a clear preference towards algorithm performance, ignoring any interpretability requirement. The person that left this comment voted "highly understandable" for every feature in any group.

Finally, we draw a few conclusions from this first part of the survey: first, the constrained feature construction algorithm (PGGGP) produces significantly more understandable features than the unconstrained (GP) version. This was quite obvious objectively since the majority of the built features with the GP algorithm do not respect the physical units, but this is now proven subjectively. No clearly visible difference appears in the understandability of features built prior to model induction or embedded into it, but this could be due to a lack of representativity notably for embedded feature construction. Finally, regular physicists' variables are probably noted slightly more understandable than automatically built features, but it must be

emphasized that the mathematical formulas of automatically built features are often more concise, which is an asset towards understandability.

8.2.4 Evaluation of the models

The same questions were asked about a GAM and a FURIA base: how a given example is classified, how to modify the model to change the classification, how sure is the respondent about his answers and a subjective evaluation of the model's transparency or ease of validation. The same questions were also asked for a neural network except for the question on how to modify the model.

8.2.4.1 GAM

Figure 8.10 displays the results of the simulation of the GAM. 64.5% of the respondents obtained the right classification and term to remove. In total, 74.2% found the right classification. The people who answered correctly were globally more sure of their answers than those who had at least one wrong answer (wrong classification or wrong term), as displayed on Figure 8.11.

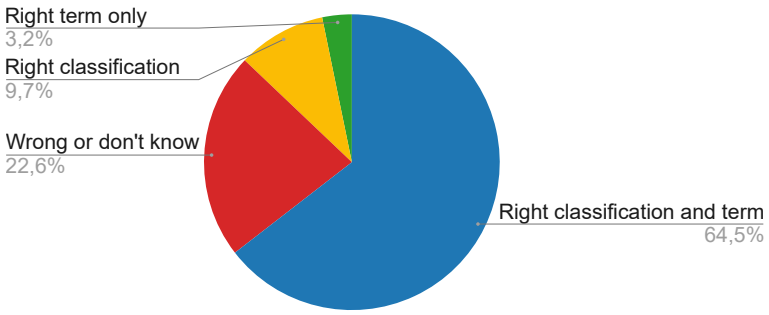


FIGURE 8.10: Results of the simulation of the GAM by the respondents.

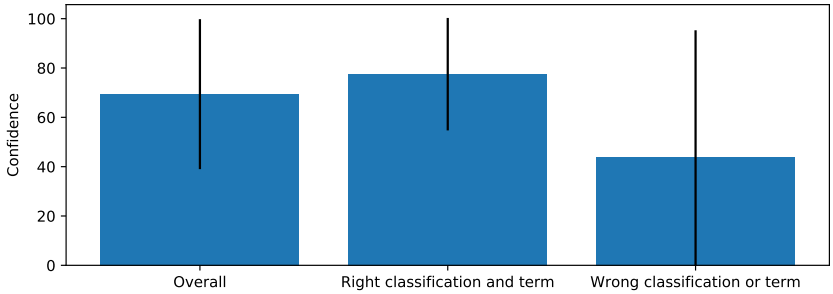


FIGURE 8.11: Respondents confidence in their answers depending on their correctness, along with the standard deviation (black line).

Globally, respondents gave an average score of 2.90 to their perceived ease of validation of the GAM. As displayed on Figure 8.12, the score is higher for people who understood well the model (3.20 in average), and lower (2.22 in average) for people who did not know how to use the model. The scores also increase with confidence in respondents' answers, as visible on Figure 8.13.

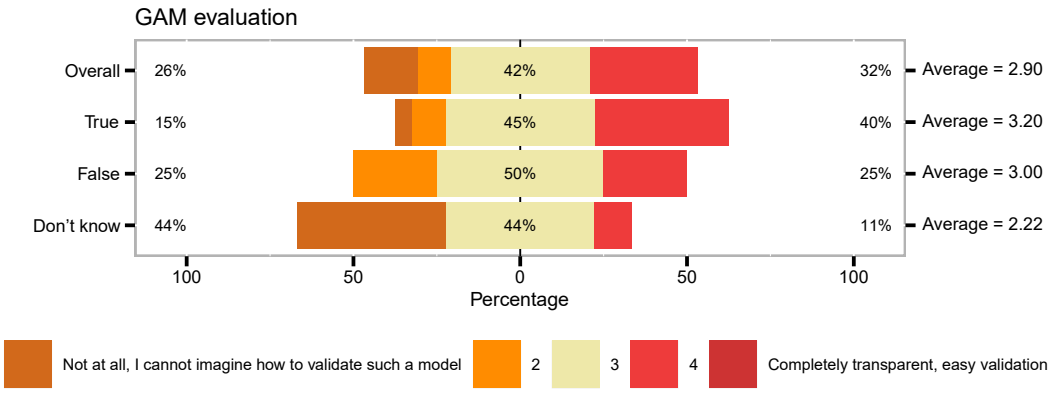


FIGURE 8.12: Transparency scores given to GAM depending on the correctness of the respondents' answers to the simulation.

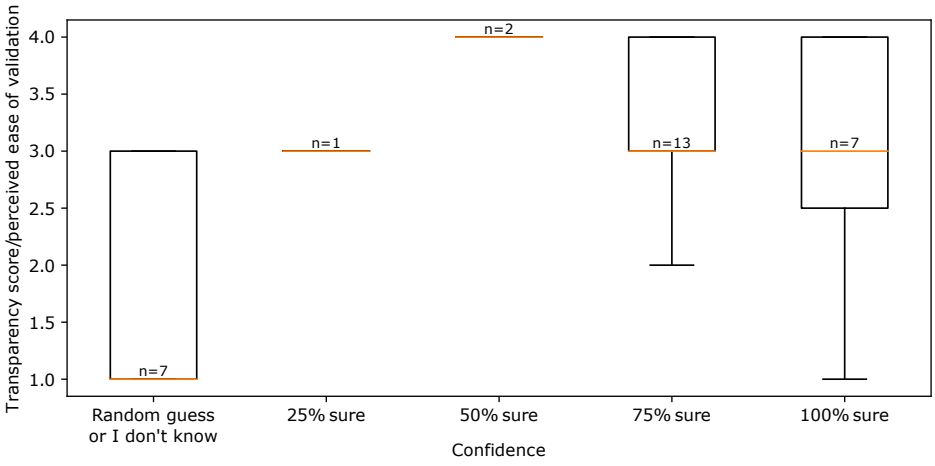


FIGURE 8.13: Transparency scores given to GAM as function of the respondents' confidence in their answers to the simulation.

The GAM received 11 comments about various subjects. Two comments add insights about the validation problem:

– "I think that the correlation between the variables X_i should be studied- I believe that this is already done. It is hardly possible to validate the model without understanding the correlation between the variables, understanding the variable is important and not essential, but in separating signal/background, it is important to understand correlations between different variables."

– "GAM do not produce really transparent results even though they are optimal, with respect to a certain training. The danger to me would be the bias introduced by the training of this model, which makes some variables more discriminant than others for a specific training set. This can be called a systematic

error... This seems to me less trustworthy than a classical analysis where this bias is more controlled, or at least understandable."

Indeed, if some variables used by the GAM are correlated, then the importance (i.e. range taken by the associated term) of these variables cannot be interpreted. About the second comment, it could be applied to other models than the GAM: biases towards a particular training set must be carefully checked. In our opinion, it should be easier with transparent machine learning models, but maybe not as simple as with cuts. This last person gave the GAM a transparency score of 1 despite correct answers to the simulation with 100% confidence.

Two respondents interestingly understood that classifying with a GAM requires choosing a threshold on the model's output:

– "How to put the frontier between a signal and background event? Is 0.5 the best choice? What is the shape of the probability distribution for a set of known signal events (and background events), is the separation completely clear (or figure of merit)?"

– "Lacking a threshold value for the classification means that the output is pure guesswork. One would (wrongfully) assume that the threshold would be halfway between 0 and 1 (so 0.5) thus possibly changing the classification by removing the second term. However, that threshold might as well be 0.9 (again, we are not told!), in which case the second does not make a difference in the classification."

Indeed, the choice of the threshold is of the first importance and will be made to optimize the physics analysis. These persons perfectly understood this challenge.

8.2.4.2 FURIA

Figure 8.14 displays the results of the simulation of the FURIA base. 50% of the respondents obtained the right classification and rule to remove, which is lower than for the GAM. Again, the confidence of the persons who gave right answers is higher than for people that are wrong (wrong classification or wrong rule to remove), as displayed on Figure 8.15.

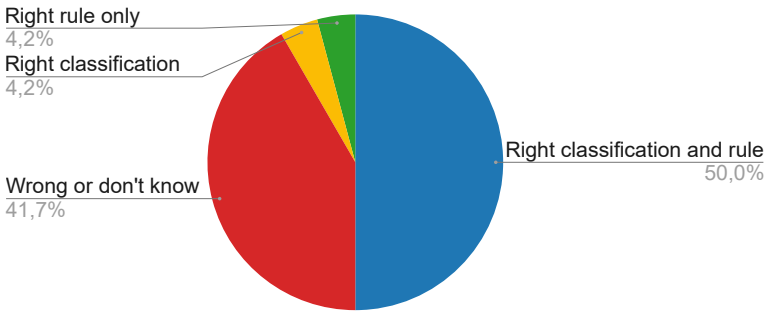


FIGURE 8.14: Results of the simulation of the FURIA base by the respondents.

Globally, the transparency of the FURIA base is noted 2.92, which is similar to the score obtained by the GAM. Again, people who did not know how to use the base gave it a lower score than others, as displayed on Figure 8.16. Interestingly, people who gave a wrong answer rated FURIA higher than people that were right for the simulation. This remark must be taken with caution: the understanding of respondents and therefore their answers and confidence highly depend on the quality of our

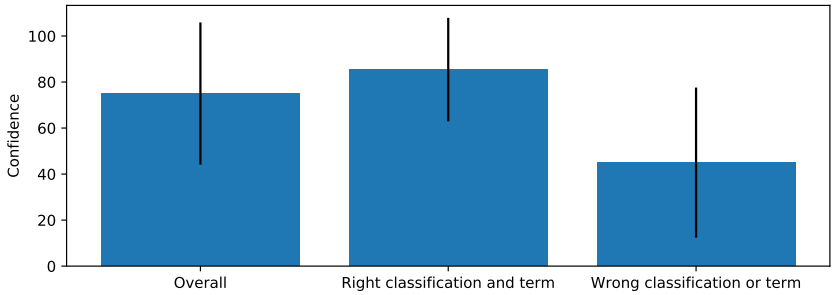


FIGURE 8.15: Respondents confidence in their answers depending on their correctness, along with the standard deviation (black line).

explanations. Some people may have thought they understood well the model and rated it high while their representation was erroneous.

The global trend correlating confidence with transparency score is visible on Figure 8.17.

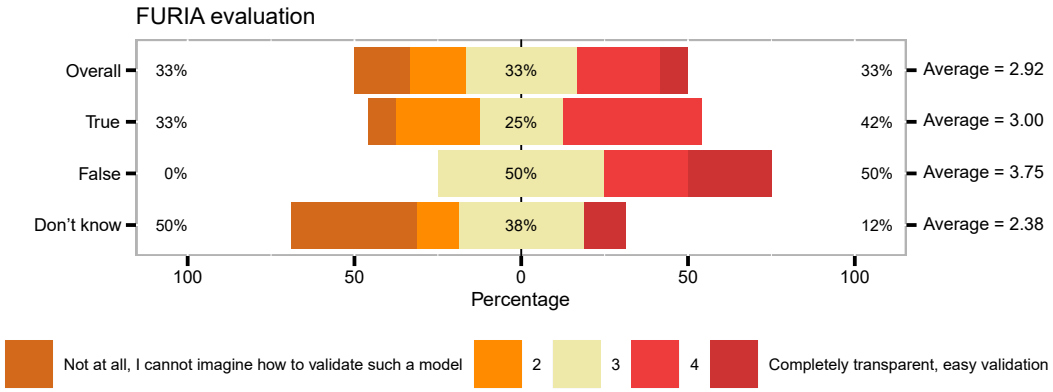


FIGURE 8.16: Transparency scores given to FURIA depending on the correctness of the respondents' answers to the simulation.

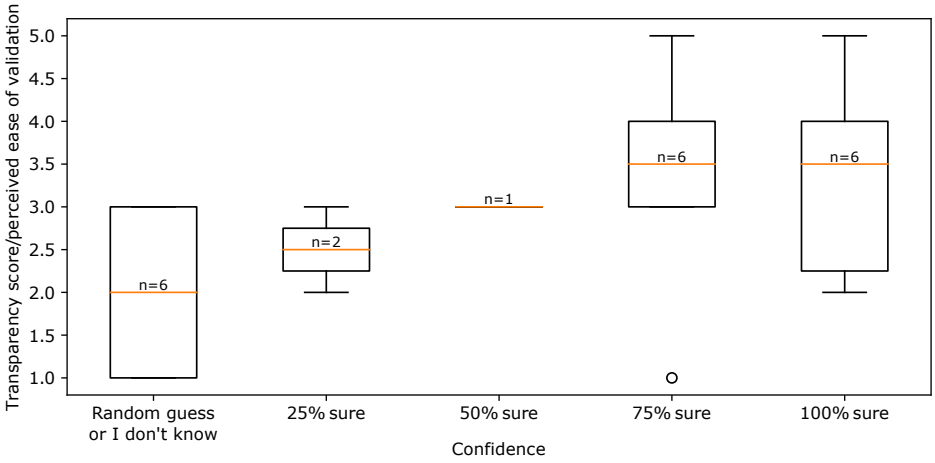


FIGURE 8.17: Transparency scores given to FURIA as function of the respondents' confidence in their answers to the simulation.

Several respondents demonstrated in the comments a high curiosity to understand how the model has been built:

- "I presume the c-values are confidence values, but no information was given on how the c-values are computed."
- "1) How are determined the numbered values indicated in the condition of the rules? (and the score?) Would it be interesting to have distributions instead of raw numbers? (a smooth real valued function (score) taking the z-momentum balance as parameter?)
  2) How are considered the correlations between rules? Can one simply sums the output of 2 correlated rules? Would it be interesting to have a smooth score function taking all the variables and constructing a score out of them?
  3) The conditions inside one rule can be correlated, and to me, can be called 'cuts'. Is this conceptually really better than a usual 'cut and count' analysis?
  4) I guess the main goal is to find the structure of the signal (or background) inside a phase space region, and try to discriminate it via a simple real valued function (score) (To integrate non linear correlation laying inside data, would it be interesting to apply a manifold learning algorithm to understand the structure of signal for instance, and put a notion of distance for one event to this manifold structure? but would it be interpretable...)
  5) Why only 5 rules for signal and 10 for π^0?"

The fact that the FURIA base comprises many rules (15 in total) disturbed a few respondents. The more rules, the more difficult it is to apprehend the model in its globality, whereas the GAM comprises only 5 terms. We were expecting that the FURIA base would be more interpretable than the GAM, but the respondents have shown themselves more severe, probably because this model is the most similar to the cuts they are used to. To conclude, one respondent preferred the GAM to FURIA:

> "More black box like than the first model."

8.2.4.3 Neural network

Finally, a few questions were asked about a neural network to make respondents more familiar with it. The respondents were supposed to use the provided SHAP values to classify the given example. However, we noticed during the response period that the description of these values could be misinterpreted. Indeed, the values taken by each variable of the given example were provided along with a color sticker indicating if the value is high or low compared to other values in the training set. These colors must be put in parallel with the SHAP diagram to determine the influence of the variables on the network's output. However, after discussing with a few respondents that left their email, we figured out that some people directly took the colors as the impact of the variable on the network's output, without using the SHAP diagram. One respondent also underlines his perplexity in the comments:

> "I did not get the model; To me the PDF seems really ill-posed... what is a SHAP value? In the PDF, next to 'SHAP value' is written 'impact on model output', this is ambiguous and can be understood as the importance of a variable in the classification (but it is not, this is for the color I guess? ...) So one should look on the variables with red dots because they are more important in the classification. OK now why do we need the values of the observables? is it the SHAP value? should we sum them weighted

with the color? I guess then most of the red variables are positive, so positive SHAP? DVCS event?"

Therefore, the results of simulation questions are skewed.

However, Figure 8.18 shows a big dispersion in people's answers, with a third of respondents giving the right classification, another third the wrong, and the last third saying they do not know how to classify the given event.

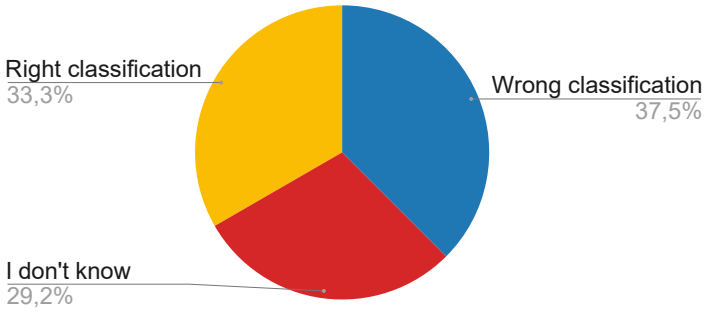


FIGURE 8.18: Results of the simulation of the neural network by the respondents.

The neural network obtains an overall transparency score of 2.08, without significant difference according to respondents' answers to the simulation (see Figure 8.19).

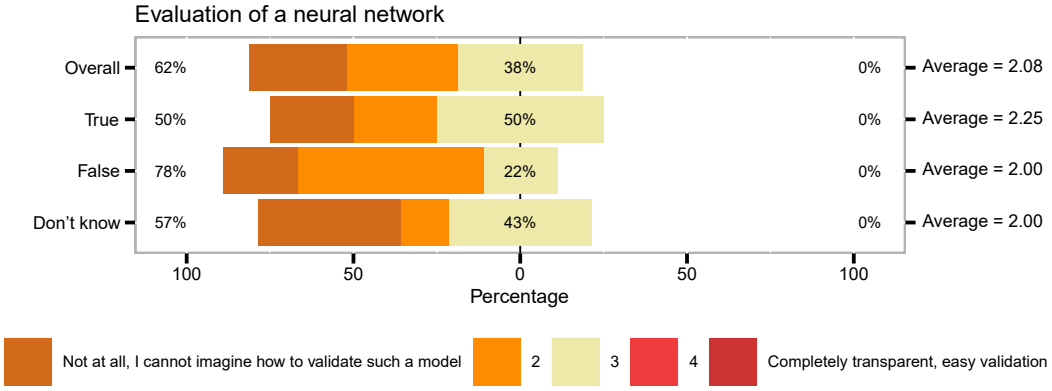


FIGURE 8.19: Transparency scores given to the neural network depending on the correctness of the respondents' answers to the simulation.

Globally, the network received more suspicion than the previous models:

- "Not enough explication given."
- "Maybe the event is more blue than red, but very difficult to conclude. You cannot convince colleagues without presenting efficiency and purity and changing the initial sample (degrading resolution or adding other type of background)."
- "I would tend not to trust a model if I do not understand the output."
- "I am not totally sure that I am interpreting the information correctly. Also, the colours are not enough to tell how far along the SHAP scale each variable is. But it does give an overall feel..."

The second comment gives an interesting point for the validation: the performance of event selection must be checked on degraded data samples. This was indeed the point

of the domain adaptation (chapter 7). Other types of background are indeed a problem here: the proposed models only classify two processes, DVCS and π^0 production. We hope the models' output for other processes will be close to 0.5, so that a proper threshold choice will be able to remove them completely. This will be discussed in chapter 9.

8.2.5 Global comparison

First, Table 8.1 summarizes the individual results of the GAM, FURIA and neural network evaluations. From these results, the GAM appears as the most interpretable of the three models: respondents have the most succeeded understanding its functioning and were the most confident about their answers. The GAM and FURIA received the same transparency score in average namely 2.9, with the neural network further behind with an average score of 2.1. This confirms that the GAM and FURIA have the potential of being understood by the respondents: some participants underlined that it would be easier for them to get the functioning of the model if given more explanations, notably on how the model was induced or the distributions of involved features.

TABLE 8.1: Transparency score, confidence and percentage of good and wrong answers for each of the proposed models.

	GAM	FURIA	Neural network
Transparency score	2.90	**2.92**	2.08
Confidence	**60.48**	56.25	39.58
Percentage of good answers	**64.50**	50.00	33.33*
Percentage of wrong answers	**12.90**	16.67	37.50*

* The description of the procedure to classify the example using the SHAP values has been misunderstood.

The last question asked to rank four event selection methods: the GAM, the FURIA base, the neural network and regular physicists' cuts. Figure 8.20 presents the counts of ranks for each model. Globally, the GAM gets the best rank (2.04 in average), but is mostly rank second, either behind the neural network or cuts. FURIA has disappointed most respondents, since it is the model ranked the most often last and third. Indeed, FURIA offers less flexibility to choose a position on the ROC curve.

– "I rate worst FURIA, because it looks the less easy to train and run."

– "FURIA is also easy to understand, but seems to be a set of fancy probability-weighted cuts and though it performs slightly better for lower π^0 contamination, it seems to be only applicable to just above 20%. I feel somehow more uneasy relying on it."

The ranks of the neural network and cuts are more disputed. Nevertheless, the neural network is the model ranked first most of the times. This comes from people preferring performance to transparency: Table 8.2 indeed presents the transparency requirements for event selection of the respondents according to the model they ranked first. 80% of people having chosen cuts as their favorite event selection technique initially required a complete understanding of the method. This proportion drops down to 33% for the neural network and for the GAM as well. Interestingly, the person who answered that

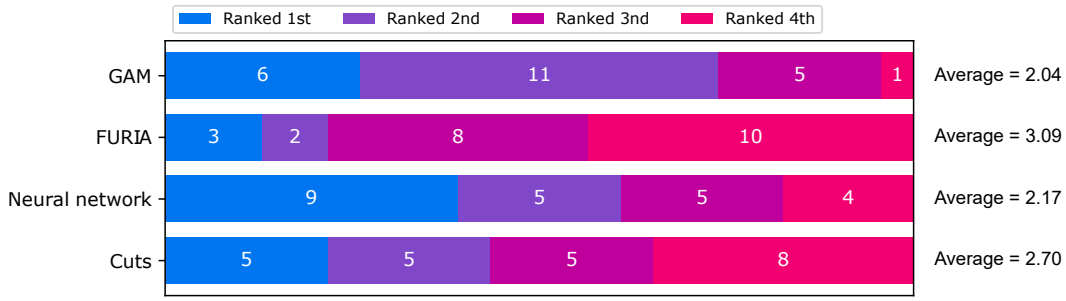


FIGURE 8.20: Ranks given to the four models, for all respondents.

there is no need for any explanation as long as the method is working well has been convinced by the GAM.

TABLE 8.2: Transparency requirements of respondents (as lines) as function of their first choice (as columns). The percentages are displayed along with the raw count between parentheses.

	Ranked GAM first	Ranked FURIA first
Sufficiently transparent	50% (3)	100% (3)
Completely transparent	33% (2)	0
No need for transparency	17% (1)	0
	Ranked neural network first	**Ranked cuts first**
Sufficiently transparent	67% (6)	20% (1)
Completely transparent	33% (3)	80% (4)
No need for transparency	0	0

If we focus on respondents that understood correctly the functioning of the GAM and FURIA, namely that they obtained correct answers to the model simulations, we obtain the ranks presented on Figure 8.21. 39% of respondents fall in that category. We observe that the ranking is less severe for FURIA, with only 6 persons that rank it last or third, against 18 initially. Similarly, the neural network looses two thirds of the persons that ranked it first.

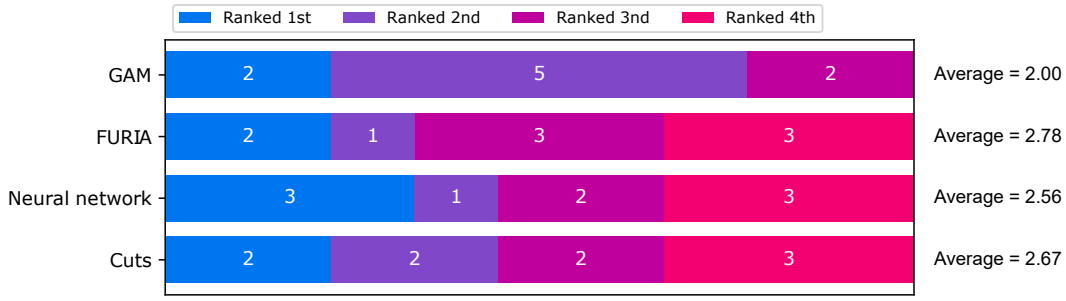


FIGURE 8.21: Ranks given to the four models, among respondents having given the right answers to the GAM and FURIA simulations.

Figure 8.22 compares the average ranks of the four models for all respondents against for respondents that gave right answers to the FURIA and GAM simulations. While

the GAM and cuts keep the same average, we observe a slight increase in FURIA rank and a slight decrease for the neural network. We can conclude that FURIA may have been misunderstood by some of the respondents, or intimidated them with its 15 rules.

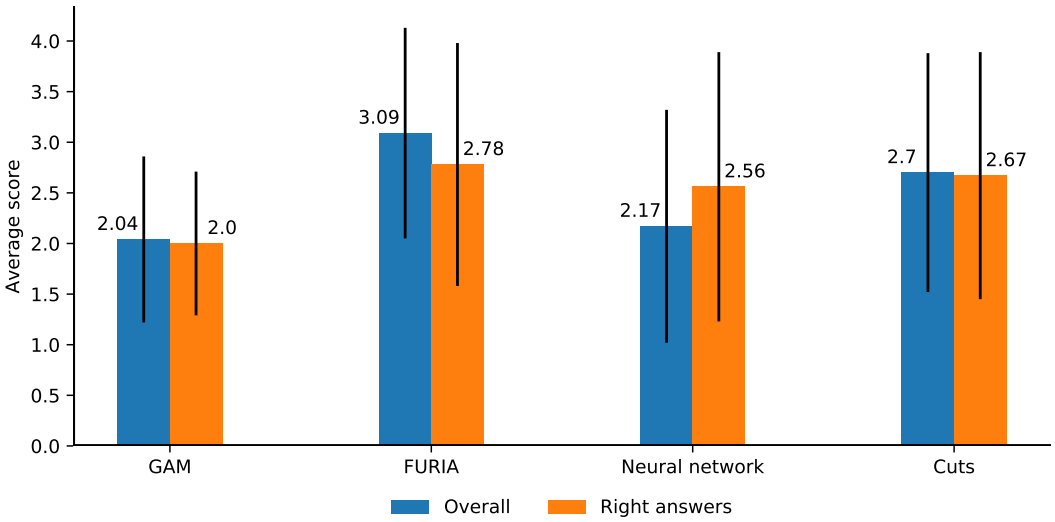


FIGURE 8.22: Comparison of average ranks given to each model for all respondents and among respondents that gave right answers to GAM and FURIA simulations. The black lines represent standard deviations.

Globally, we can say that the GAM has been well appreciated by the respondents.

> "GAM is easy to understand and very transparent and seems robust."

FURIA is controversial. For some people, it looks like common physicists' cuts and therefore easier to understand, for others the rules are too numerous and complicated. Moreover, it offers less flexibility for efficiency/contamination balance.

> "To me, all the methods are far more complicated to implement than a usual 'cuts' analysis. FURIA provides interesting improvements to be considered."

The usual cuts received various ranks. Some respondents underlined that a ROC curve could be obtained by varying the values of the cuts, which is true but more complicating since it involves multi-dimensional cuts instead of a single cut on the output of a machine learning model.

- "For the 'cuts' model you give only one point, i.e. a specific set of cuts. But in principle, one can study the signal and background fraction when varying the cuts."
- "For the cuts model it would have been informative to vary the size of the cuts to get a dependence rather than a single point. One suspects that with very generous cuts one will pick up all DVCS events while not incurring (at least not according to the plot) more than 50% π^0 background."

Finally, respondents either ranked first the neural network because of its best classification performance or were suspicious about it:

- "The neural network is the best working model, but 1) it's a completely black box 2) I did not understand your explanation 3) people not familiar with them, dislike them."
- "Although the neural network is the most efficient I cannot imagine its behavior when introducing noise and smearing... They can be perfect for other tasks than data selection though..."
- "Neural networks clearly show the best performance, but, based purely on the info in this survey, I do not feel I am confident enough understanding how to use the information. This can be fixed with a more in-depth study, though, so I might change my preference. For a PhD student or a postdoc just starting out, GAM might be a clearer starting point."

To conclude on the model comparison, FURIA is too complex to apprehend globally and its performances are not satisfying enough according to the physicists who answered. This conclusion permits to be cautious as well for decision trees: they are often more complex than FURIA bases, although their performance is better and they do not suffer from the lack of flexibility of FURIA. For the physics analysis, we should settle for a small decision tree, not deeper than 3 or 4 layers. A GAM with only 5 terms approaches very closely the performance of the neural network, which makes it a very interesting compromise for event selection.

8.3 Conclusion

In this chapter, we proposed an interpretability survey to assess the subjective validation of the developed transparent models by physicists. We designed a survey to evaluate the proper understandability of the automatically built features on the one hand, and to evaluate the interpretability of two transparent models, a FURIA base and a GAM, on the other hand. Finally, a global comparison between these two models, a neural network and standard cuts was conducted.

This study on the physicists' perception of transparent machine learning models have led to very interesting results. We confirmed the understandability of the features built by automatic constrained feature construction. The transparency of a GAM model and a FURIA base has also been validated, the GAM having received very positive comments. We also studied the subjective preference of respondents after presenting them several models for event selection. From the results of the survey, we discovered that the respondents globally preferred the GAM over FURIA, and that the GAM is a very good candidate to replace the usual cuts in a physics analysis. We should pay attention to the size of the proposed models: the number of rules of FURIA may have startled a few respondents, while the first 5 terms of the GAM suffice to obtain a satisfying classification performance.

All results and comments can be found in Appendix G. We would like to warmly thank all participants to the survey and in particular the six first testers: Francesco Bossu, Nicole d'Hose, David Lhuillier, Loïc Thuilliez, Marine Vandebrouck and Michael Winn.

Chapter 9

Analysis of DVCS data from CLAS12

9.1 DVCS event selection in CLAS12 data 186
9.1.1 Momentum corrections 186
9.1.2 Machine learning models for the analysis 188
9.1.3 Removal of other background processes 189
9.2 π^0 subtraction and asymmetry computation 190
9.2.1 Principle 191
9.2.2 Results and validation of the proposed models 192
9.3 Optimal selection threshold and asymmetry computation 194
9.3.1 Optimal selection threshold to minimize the asymmetry statistical error 194
9.3.2 Asymmetry computation 196
9.4 Comparisons with other techniques 197
9.4.1 Impact of momentum corrections and domain adaptation . 197
9.4.2 Alternative methods for event selection 199
9.5 Conclusion and perspectives 201

Data acquisition at CLAS12 for DVCS on an unpolarized target started in spring 2018 and is divided into several periods:

- The first period of spring 2018 encountered a few difficulties due to unoptimized detector settings and high background noise. The reconstruction is therefore challenging for this period;
- The setup was improved for the fall 2018 data taking. Reconstruction for this period has been successfully completed;
- Data taking continued in spring 2019 with a beam energy lowered to 10.2 GeV. Reconstruction is currently ongoing.

For this analysis, we use data from fall 2018, which constitutes over 1 Po of raw data. During this period, two settings varied: the beam helicity (plus or minus 1) and the torus magnetic field orientation (leading to electron trajectory bent inwards or outwards of CLAS12). Comparing the yields for the different beam helicities permits to compute the beam spin asymmetry (see section 9.3). The inbending setup naturally cuts electrons leading to $Q^2 \leq 1.5$ GeV^2/c^4 since these electrons have a small scattering angle and are sent towards the beamline by the torus magnetic field. On the

opposite, the outbending configuration will get a large number of these small scattering angle electrons, but many protons will be lost when directed towards the forward detector since their trajectories will be deviated towards the beamline.

Preselection cuts are applied on the data to ensure its quality:

- the vertex position of the electron and proton must be between -8 and 5 cm along the z-axis;
- the χ^2 of particle identification for the electron and proton must be below 3;
- the angle between the electron and the photon must be over 5°;
- the photon energy must be over 0.5 GeV;
- β of the photon must be comprised between 0.9 and 1.1;
- the polar angle θ of the photon must be over 2.75°;
- fiducial cuts (i.e. geometrical cuts to remove the edges of the detectors) are applied on the drift chambers (forward detector), forward calorimeter and forward tagger calorimeter.

For DVCS event selection, additional loose cuts are performed to remove the majority of background:

- the missing energy $ep \rightarrow ep\gamma X$ must be below 1.4 GeV in absolute value;
- the squared missing mass $ep \rightarrow ep\gamma X$ must be below 0.2 $\mathrm{GeV}^2/\mathrm{c}^4$ in absolute value.

The work presented in this chapter has been carried out in close collaboration with Maxime Defurne from CEA Irfu and Guillaume Christiaens from the University of Glasgow.

9.1 DVCS event selection in CLAS12 data

9.1.1 Momentum corrections

As detailed in the experiments of chapter 7, the proposed domain adaptation technique is only applicable in the region of the phase space where exclusive π^0 production events are found in abundance, namely in the two kinematic bins:

- $0.16 < x_B < 0.26$ and $Q^2 < 2.4\,\mathrm{GeV}^2/\mathrm{c}^4$;
- $0.26 < x_B$ and $Q^2 < 3.25\,\mathrm{GeV}^2/\mathrm{c}^4$.

For the physics analysis, 24 bins are considered in total over Q^2, x_B and t, displayed on Figure 9.1. The two regions of the phase space where domain adaptation can be applied group the bins 2, 5, 8, 11 and 3, 6, 9, 12. The four bins in each region correspond to subdivisions in t.

In the 16 remaining bins, the domain adaptation technique is not applicable as such. Instead, a backup solution is to perform momentum corrections: the idea is to correct manually the variables that are the most biased. We know this is currently the case for the photon energy, which is underestimated by the calorimeters and for the proton in the central tracker, which suffers from calibration errors. Figure 9.2 displays the squared missing mass $ep \rightarrow e\gamma\gamma X$ for exclusive π^0 production events in Monte Carlo

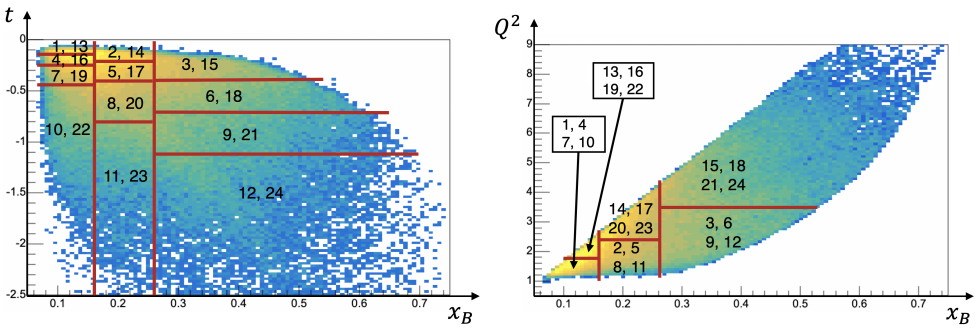


FIGURE 9.1: Kinematic bins used for the physics analysis. Left: in the (x_B, t) plane; right: in the (x_B, Q^2) plane. The numbers identify the bins [Christiaens, 2021].

simulated data (with cross-sections) and in CLAS12 data, for photons in the forward tagger (scattering angle θ below 5°) or in the forward detector ($\theta > 5°$). There is a clear shift of the distribution in particular regarding the forward tagger, while the peak of the distribution should be at the squared mass of the proton namely 0.88 $\mathrm{GeV}^2/\mathrm{c}^4$.

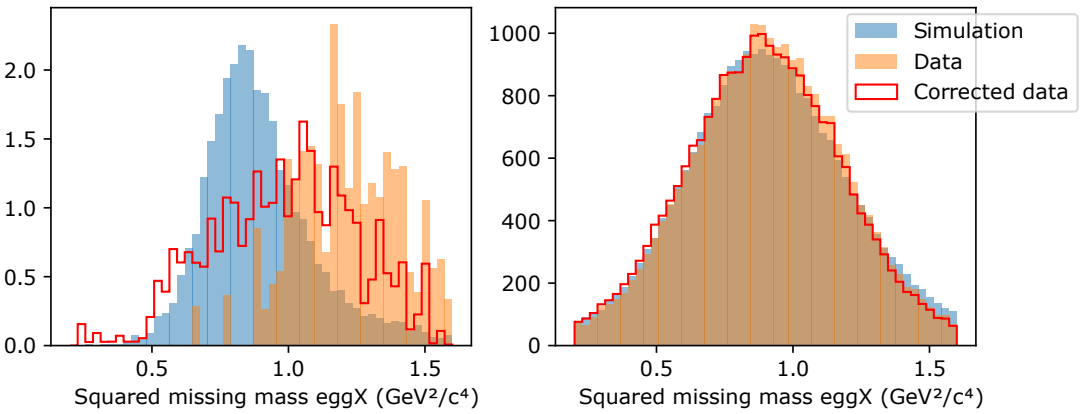


FIGURE 9.2: Squared missing mass $ep \to e\gamma\gamma X$ for exclusive π^0 production events in Monte Carlo simulated data and in CLAS12 data. Left: photons were sent in the forward tagger ($\theta < 5°$); right: in the forward detector ($\theta > 5°$).

Since calibration is still ongoing for the central tracker, the proton is not reconstructed properly. Therefore, momentum corrections are applied to both the photon energy and the proton three-momentum. They are performed using exclusive DVCS events, selected with strict cuts, with π^0 subtraction as explained in 9.2. For these corrections, the hypothesis is made that the electron and the direction of the photon are well reconstructed. This information suffices to recompute the four-momenta of the particles in the output state and develop a momentum correction strategy for both the photon energy and proton three-momentum.

The distributions of the squared missing mass with momentum corrections are displayed in red on Figure 9.2. In the remaining of this chapter, corrected data will be used for the analysis unless otherwise stated.

9.1.2 Machine learning models for the analysis

In the end, a specific domain adaptation should be performed using the corrected data as target, with events covering the entire phase space as discussed in chapter 7. In practice, in this analysis, we use domain adaptation in bins where it is possible (8 bins over 24), and momentum corrections elsewhere. We also compare the analysis using momentum corrections against the analysis with domain adaptation on the eight compliant bins.

Three models are considered in the following:

- A FURIA base with prior feature construction of 5 features (the same that was studied in the interpretability survey of chapter 8). It comprises 11 rules.
- A fuzzy C4.5 Fibo with embedded feature construction of 15 features. It has 95 non-leaf nodes but we will try limiting its maximal depth to 3 and see the impact on the analysis.
- A FCGAM_b_{max} with 16 terms (the same that was used for domain adaptation in chapter 7, and this model restricted to 5 terms was also used in the interpretability survey of chapter 8). We will try reducing the number of terms to 5 as in the survey and see the impact on the analysis as well.

In addition, we consider a neural network and physicists' cuts (details respectively in 9.4.2.2 and 9.4.2.1), as competitor models.

The ROC curves and efficiency against contamination curves of these models on Monte Carlo simulated data with cross-sections are displayed on Figure 9.3. However, these classification performances are probably overestimated and we do not know them precisely on real data. At first sight, the relative ordering of the three best models reverses in the middle: the neural network is the best model in terms of classification performances at low contamination, followed by the FCGAM and the fuzzy C4.5. However, at high efficiency (i.e. true positive rate), the fuzzy C4.5 has lower false positive rates, followed by the GAM and the neural network. FURIA permits to retain more than twice as much DVCS events compared to the cuts while keeping approximately the same contamination level.

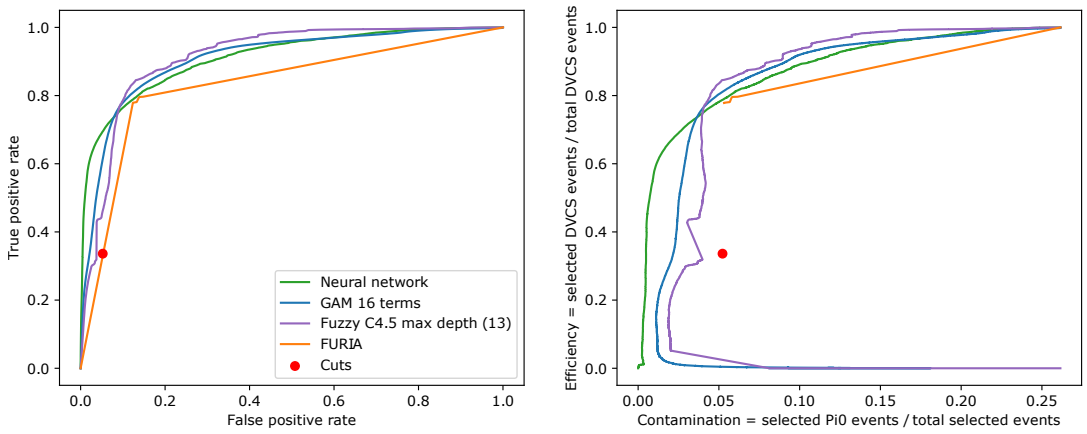


FIGURE 9.3: ROC curves (left) and efficiency against contamination curve (right) for the considered models.

9.1.3 Removal of other background processes

Simulation data used for training machine learning models consist exclusively of DVCS and π^0 production events. Therefore, there is no third class for other background processes. However, other processes may contaminate the event selection despite π^0 production is the main background for DVCS.

Intuitively, other processes should get an output around 0.5, namely the classification is not sharp since they are neither π^0 production events nor DVCS events. To remove them, a higher threshold must be determined.

As control process, we use η production. η is a meson 5 times heavier than the π^0 meson. It decays mostly into two photons (40% of cases), three π^0 (30%) or $\pi^+ + \pi^- + \pi^0$ (22%) [Zyla et al., 2020]. The non-exclusive π^0 should be removed thanks to π^0 subtraction. However, the two photons channel may constitute a contamination to DVCS selection for the same reasons than for exclusive π^0 production, considering the energy resolution for the photons. However, the invariant mass of these two photons equals the mass of the η instead of the π^0 mass, as displayed on Figure 9.4.

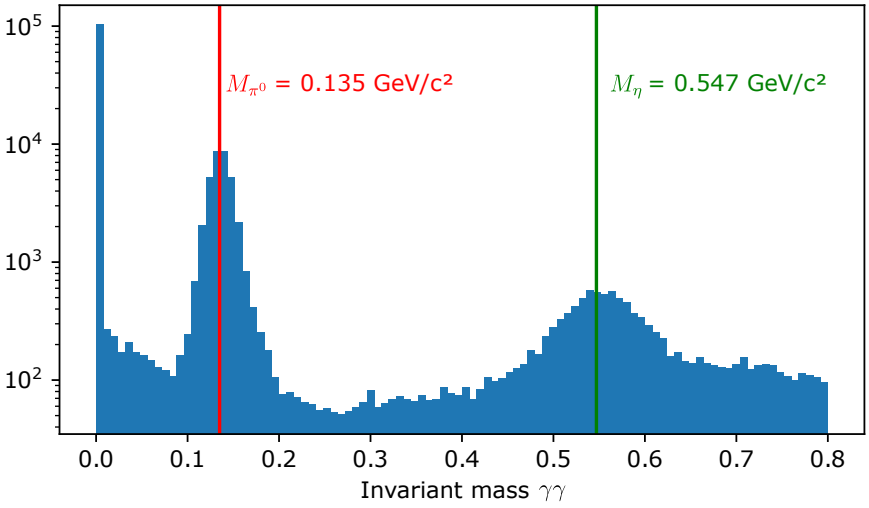


FIGURE 9.4: Invariant mass $\gamma\gamma$ in CLAS12 data after applying cuts on the total missing energy and squared total missing mass. The presence of π^0 production events and η production events is clearly visible.

Two-photons η production events are isolated in the data using the following cuts:

- the missing energy $ep \to ep\gamma\gamma$ must be below 1.8 GeV in absolute value;
- the squared total missing mass $ep \to ep\gamma\gamma$ must be below 0.2 GeV^2/c^4 in absolute value;
- the invariant mass $\gamma\gamma$ must be between 0.502 and 0.592 GeV/c^2 (0.045 GeV/c^2 apart from the η mass).

5230 η production events are retrieved in this way. Applying the selected machine learning models on these events with momentum corrections produces Figure 9.5. FURIA is not displayed on Figure 9.5 since the large majority of events (86%) are classified to 0 or 1. More precisely, 13% of η production events are classified as 1 by FURIA.

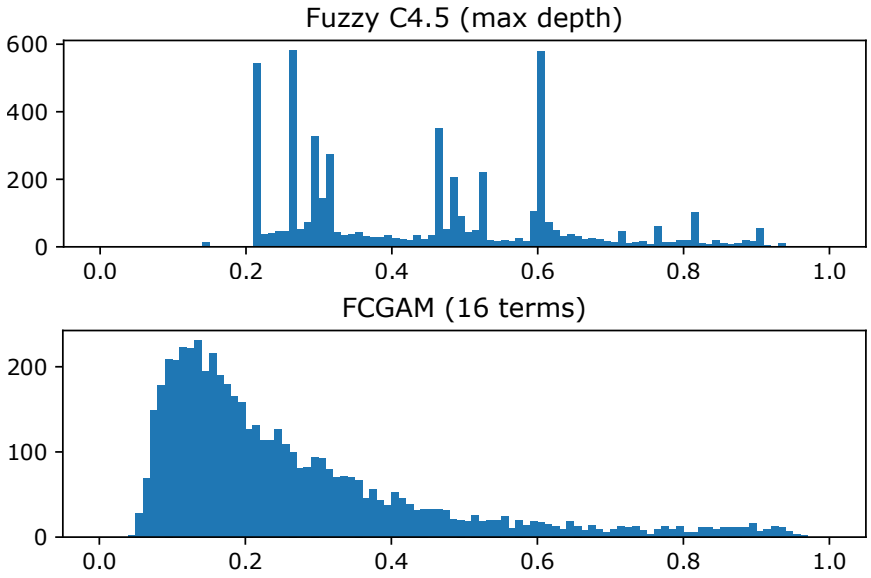


FIGURE 9.5: Output of machine learning models for η production events in data. 1 corresponds to DVCS and 0 to π^0 production.

These plots give an estimate of the minimal threshold on the models' outputs to eliminate the majority of η production events. Starting from 0.6 – 0.65, η production events and hopefully other remaining background events should be absent from the selection. Another approach would be to perform an η subtraction similar to the π^0 subtraction detailed in 9.2, therefore enabling lower selection thresholds. This would also require ensuring that no other background process is significantly contaminating the selection.

For this analysis, to make sure to remove the majority of η production background, we set the minimal selection threshold so that 95% of the η background is removed, provided there is a significant proportion of η production events in data. This minimal selection threshold is determined per kinematic bin on Q^2, x_B, t and varies depending on the studied classification model. In some kinematic bins, almost no η production event is present. Therefore, we set the minimal threshold to 0.6 as a precaution, to eliminate any other background process. For FURIA, we keep only the events classified as 1. Analysis results using FURIA should be taken with precaution since many η production events remain in the selected sample.

Considering the entire phase space, distributions of the squared missing mass $ep \to ep\gamma X$ and of the photon cone angle for different selection thresholds of the FCGAM are displayed on Figure 9.6. Below 0.6, more background is present, notably η production events. This can be seen notably on the cone angle, expected to be 0 for DVCS: the peak progressively moves towards lower values, suggesting that additional background is present at thresholds 0.4 and 0.6 compared to thresholds 0.8 and 0.9.

9.2 π^0 subtraction and asymmetry computation

If we consider that all other backgrounds have been removed, the only remaining background is π^0 production. This section presents a methodology to remove this background and choose the optimal selection threshold to minimize the statistical error of the asymmetry.

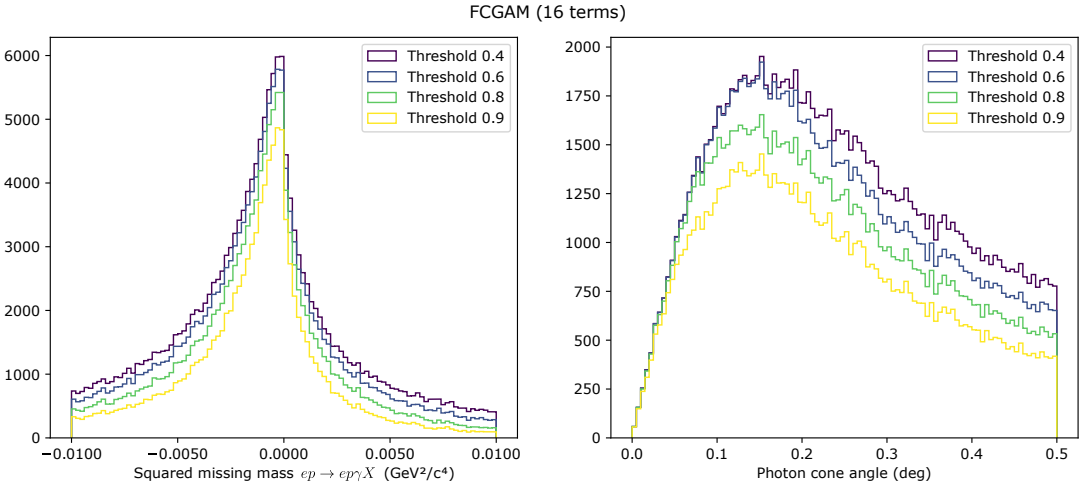


FIGURE 9.6: Distributions of the missing mass $ep \to ep\gamma X$ and of the photon cone angle for different selection thresholds on the models.

9.2.1 Principle

The different steps to perform π^0 subtraction are summarized on Figure 9.7.

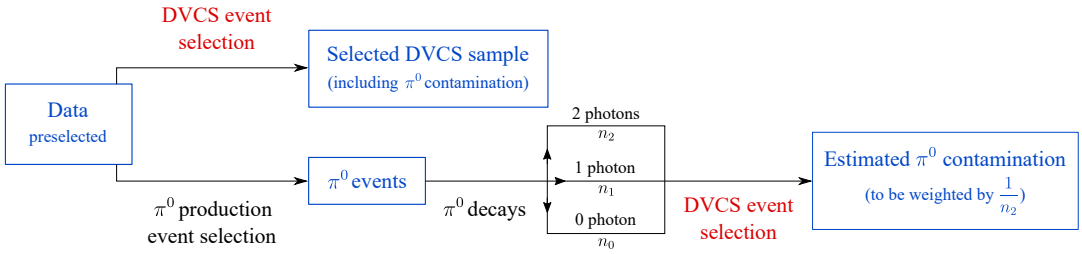


FIGURE 9.7: π^0 subtraction principle.

As stated before, π^0 production events are easy to isolate in data, as long as the two photons emitted from its decay are detected. Keeping only events that present an electron, a proton and at least two photons and that satisfy a few conservation laws permit to constitute a sample of π^0 production events of great purity. The cuts to retrieve two-photons π^0 events are the following:

- the missing energy $ep \to ep\gamma\gamma$ must be below 1.8 GeV in absolute value;
- the squared total missing mass $ep \to ep\gamma\gamma$ must be below 0.2 GeV^2/c^4 in absolute value;
- the invariant mass $\gamma\gamma$ must be between 0.09 GeV/c^2 and 0.18 GeV/c^2.

Note that both exclusive and non-exclusive π^0 production events are selected with this set of cuts. Indeed, all production channels constitute a contamination to the DVCS. Applying these cuts on reconstructed data from fall 2018 gives 260291 two-photons π^0 production events for inbending data and 483067 for outbending data.

Then, the π^0 four-momentum is computed as the sum of the two detected photons for each event. Each π^0 is decayed $N = 1500$ times into two new photons. The decays are isotropic in the π^0 frame. Then, fiducial cuts are applied on the photons to determine if they would be detected by the calorimeters and preselection cuts detailed in the introduction of this chapter are applied as well.

Over all decays in one bin, n_0 is the number of times no decayed photon is detected, n_1 the count when only one photon passes the selections and n_2 when the two photons are detected and pass the selections.

All events resulting from the π^0 decays are submitted to the DVCS event selection: this can be done either with machine learning models or using standard exclusivity cuts, as long as the method is consistent with the one used for the actual event selection. The events that pass the DVCS selection constitute the π^0 contamination. Since only two-photons π^0 production events were selected initially to perform this contamination estimation, contaminating events must be weighted by $\frac{1}{N}\frac{N}{n_2} = \frac{1}{n_2}$ when subtracted from the selected DVCS events.

Sometimes, the ratio $\frac{n_2}{N}$ is not big enough to ensure a proper contamination estimation, namely the two photons decayed out of the π^0 are almost never detected. In this case, the π^0 event is rejected. Conversely, DVCS events for which π^0 subtraction is not possible must be rejected as well. To this end, the photon of a DVCS event is artificially converted into a π^0 with same momentum and decayed $N = 1500$ times. Over these decays, if $n_2 < 0.04\,N$ (the two photons are detected in less than 4% of cases) then the corresponding DVCS event is rejected.

The performance of this procedure requires retrieving the full set of π^0 production events in data that have two detected photons. Otherwise, the n_2 weighting is overestimated and therefore the contamination is underestimated.

9.2.2 Results and validation of the proposed models

Figure 9.8 displays the models' outputs for the preselected data as well as the estimated π^0 contamination. It must be noted that the contamination is sometimes overestimated (notably at low output), thus it is not excluded that it is over or underestimated elsewhere. The implementation of π^0 subtraction is still ongoing work. Again, FURIA classifies the vast majority of events to 0 or 1.

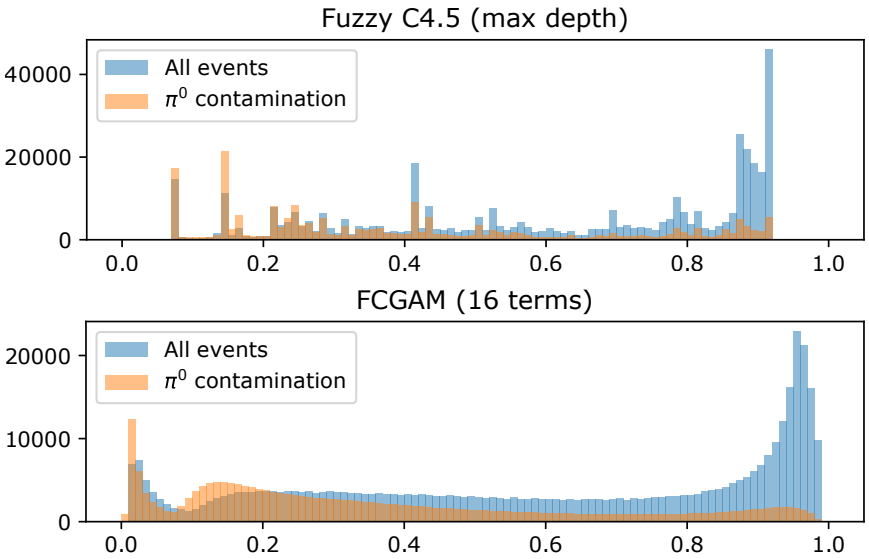


FIGURE 9.8: Outputs of classification models for preselected data and estimated π^0 contamination.

We can use the π^0 subtraction process to validate the functioning of the proposed models. Figure 9.8 already confirms that π^0 production events are classified closer

to 0 and DVCS events closer to 1 for the FCGAM and the fuzzy C4.5. Concerning FURIA, the large majority of events is again classified as 0 (90%). As seen on the ROC curves (Figure 9.3), FURIA has a limited flexibility for choosing the output threshold. Varying it should not alter significantly the results of the analysis. Retaining exclusively events classified as 1 leads to 24% π^0 contamination in the selected set, comprising 272224 events.

The five first terms of the FCGAM model can be found at the end of Appendix G. Figure 9.9 displays the distributions in data of two first features used by the FCGAM and the associated terms, stacked with the induced shape function. The distributions of the two variables are compliant with the shape functions. The first feature has been seen multiple times in the experiments in part II: it is the sum of the momenta of the three particles of DVCS along the beam axis. For this first feature, it is more likely to have a π^0 production event for low values of this feature, and conversely. However, the range taken by the shape function is much larger than the values taken by data. Either preselection cuts drastically reduced this range (especially the missing energy), or the training dataset covered a larger phase space than the data. The second feature compares the two photons produced in the collision, probably to detect π^0 production events. It apparently presents a peak around 12 – 15°, which is consistent with the induced shape function.

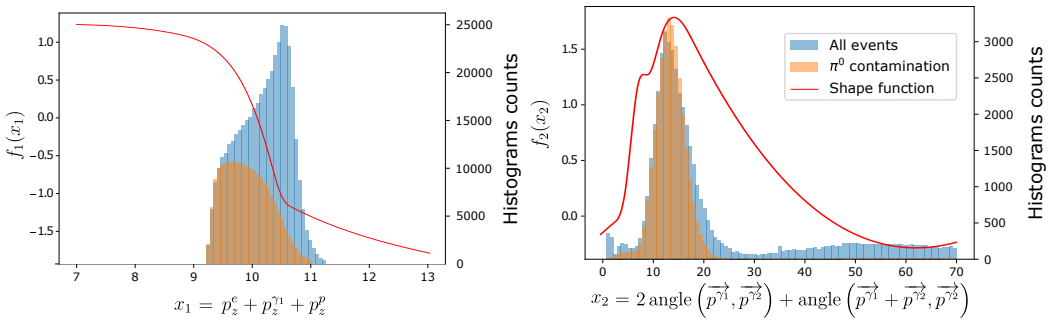


FIGURE 9.9: Distributions of the two first variables used by FCGAM. The induced shape functions (in red) are superimposed. Attention: the shape functions are reversed here. Indeed, the smaller the value taken by the shape function, the most likely the event is classified as DVCS, and conversely.

Figure 9.10 displays the distributions of the first three features used by the fuzzy C4.5 (at the root and its two children nodes), along with the fuzzy cut. The first feature is a momentum conservation check for DVCS. It perfectly corresponds to what is expected: the cut on this variable isolates more DVCS events to the left, and more π^0 production events to the right side of the threshold. The second variable (corresponding to the left branch of the root node, with a higher DVCS purity) compares the electron with the photon, considering notably their transverse momenta. The electron and the photon account indeed for most of the energy balance. The second variable separates more π^0 production events to the left of the induced fuzzy threshold. Finally, the third variable (corresponding to the right branch of the root node) involves the first photon with several of its features, and notably compares the azimuthal angles of the first and second photons. This feature probably aims at recognizing π^0 production events. It improves the purity of the sample in its right side (greater than the threshold), which contains a larger proportion of π^0 production events than the left side.

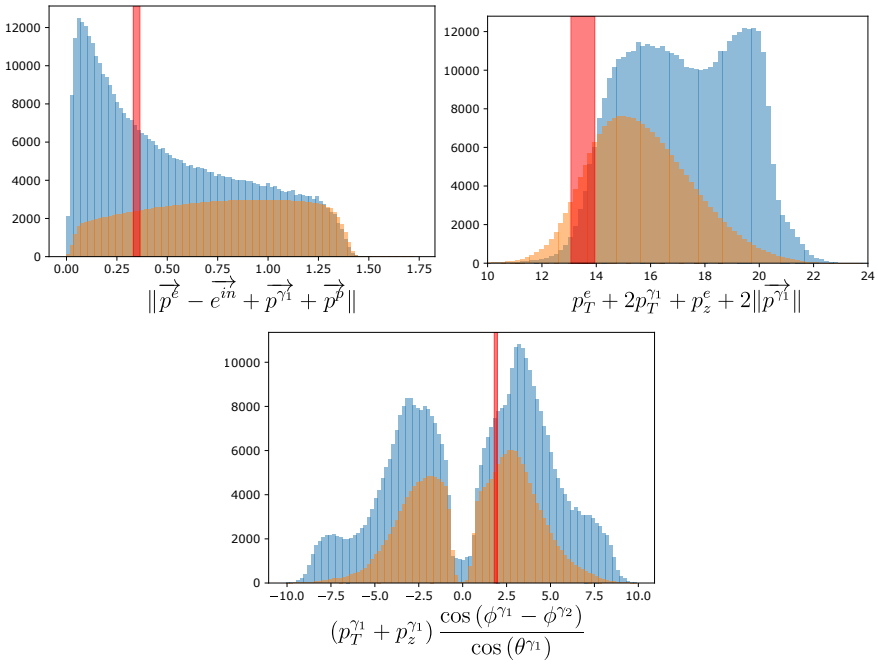


FIGURE 9.10: Distributions of the first three variables used by the fuzzy C4.5. The fuzzy thresholds (in red) are superimposed. The first cut was not applied to produce the distributions of the two next cuts.

9.3 Optimal selection threshold and asymmetry computation

9.3.1 Optimal selection threshold to minimize the asymmetry statistical error

The final selection threshold is determined to minimize the statistical error on the asymmetry. The asymmetry and its statistical error write:

$$\mathcal{A} = \frac{1}{P}\frac{N^+ - N^-}{N^+ + N^-}, \tag{9.1}$$

$$(\Delta\mathcal{A})^2 = \frac{(\Delta N^+)^2(2N^-)^2 + (\Delta N^-)^2(2N^+)^2}{P^2(N^+ + N^-)^4}, \tag{9.2}$$

with P the polarization degree (between 0 and 1), N^+ the number of selected events at helicity 1 and N^- the number of events at helicity -1. Whatever the helicity, the number of events we consider is the subtraction of the number of selected events with the quantity of estimated π^0 contamination. Namely:

$$N = N_{data} - N_{cont} \tag{9.3}$$

$$\Delta N_{data} = \sqrt{N_{data}} \tag{9.4}$$

$$N_{cont} = \sum_{cont} \frac{1}{n_2} \tag{9.5}$$

$$(\Delta N_{cont})^2 = \sum_{cont} \left(\frac{\sqrt{n_2}}{n_2^2}\right)^2 \tag{9.6}$$

$$(\Delta N)^2 = (\Delta N_{data})^2 + (\Delta N_{cont})^2 \tag{9.7}$$

We perform the computation of the optimal threshold per kinematic bin. In each bin, a minimal selection threshold is set to remove 95% of η production events provided the ratio $\frac{N_\eta}{N_{data}}$ is over 1%.

For instance, Figure 9.11 displays the computation of the optimal threshold for the FCGAM in bin 6. This bin comprises a lot of π^0 production events, therefore the contamination is high. Rather than minimizing the asymmetry error, it is rather the removal of η production events that determines the selection threshold. The asymmetry is computed considering the events with $\phi \in [72°,108°]$.

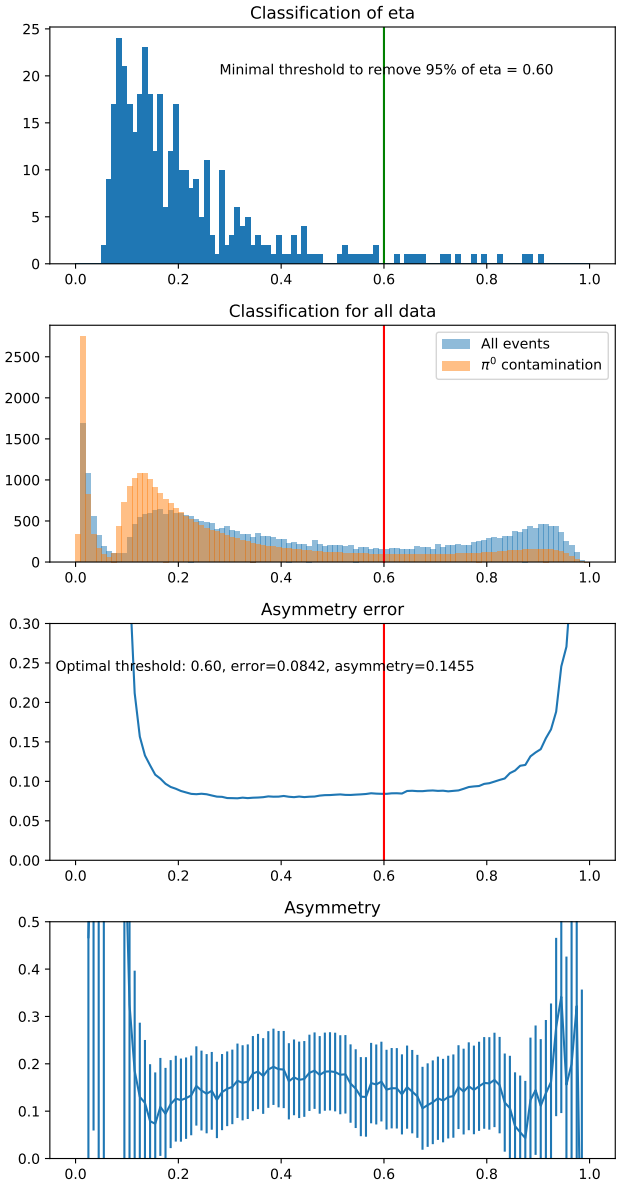


FIGURE 9.11: Determination of the optimal selection threshold for FCGAM in bin 6. The green vertical line materializes the minimal acceptable selection threshold and the red vertical line the optimal threshold above this limit. The error and asymmetry have been computed with ϕ between 72° and 108°, while the two top plots comprise the whole range in ϕ.

On the opposite, Figure 9.12 displays the same computation but for the fuzzy C4.5

in bin 13, where there is almost no contamination. Therefore, the asymmetry error increases simply with the selection threshold.

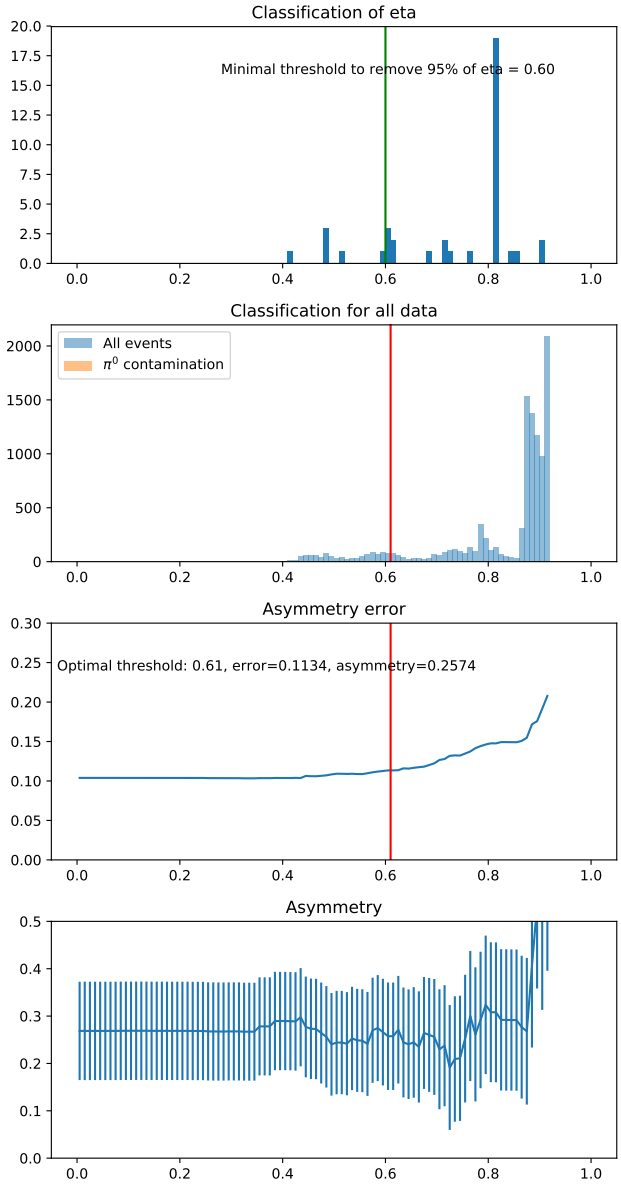


FIGURE 9.12: Determination of the optimal selection threshold for the fuzzy C4.5 in bin 13. The green vertical line materializes the minimal acceptable selection threshold and the red vertical line the optimal threshold above this limit. The error and asymmetry have been computed with ϕ between 72° and 108°, while the two top plots comprise the whole range in ϕ.

9.3.2 Asymmetry computation

Finally, the asymmetries are computed for each kinematic bin. The asymmetries for the 24 bins using the FCGAM are displayed on Figure 9.13. From a theoretical viewpoint, the asymmetry should be a function of the kinematic variable ϕ of the form:

$$\mathcal{A} = \frac{A \sin\phi}{1 + B\cos\phi}. \tag{9.8}$$

This description matches what is observed on Figure 9.13. In addition, the results of the fits of parameters A and B are printed in each bin below the asymmetry curve.

Asymmetries plots can be found for other models in Appendix H. The results are consistent while using different event selection techniques, which permits to globally validate all models simultaneously.

9.4 Comparisons with other techniques

9.4.1 Impact of momentum corrections and domain adaptation

By default, we use the momentum corrections designed by Maxime Defurne for the analysis. However, we compare here the results of the analysis with those that would be obtained with the original data without momentum corrections, and with those that would be obtained with the transferred and retrained FCGAM and fuzzy C4.5 using the domain adaptation of chapter 7. Table 9.1 compares a few metrics over bins 2 and 3 (both are comprised in the subspace covered by domain adaptation): the statistics (i.e. number of selected events), the percentage of π^0 contamination and the asymmetry statistical error on the parameter A of the fit.

TABLE 9.1: Statistics and contamination percentage in two bins with or without momentum corrections and considering domain adaptation.

	Bin 2			Bin 3		
	Stat.	Cont.	Error	Stat.	Cont.	Error
FURIA without corrections	7920	**11.6%**	0.025	19379	**37.2%**	0.027
FURIA with corrections	**8148**	12.1%	0.025	**20211**	37.6%	**0.025**
FCGAM without corrections	6528	9.1%	0.028	14528	29.5%	0.029
FCGAM with corrections	**6648**	9.1%	**0.027**	**15743**	30.3%	**0.027**
FCGAM transferred	3594	6.0%	0.035	2346	**13.4%**	0.068
FCGAM retrained	2569	**5.1%**	0.042	6263	19.0%	0.041
Fuzzy C4.5 without corrections	5308	8.8%	0.030	14956	32.4%	0.029
Fuzzy C4.5 with corrections	5588	9.1%	0.030	**15293**	**32.0%**	**0.028**
Fuzzy C4.5 transferred	**5990**	10.0%	0.030	14621	32.7%	0.030
Fuzzy C4.5 retrained	5695	**8.5%**	0.030	14301	33.3%	0.031

Overall, the momentum corrections are efficient: they permit to diminish the asymmetry statistical error compared to the error without corrections, in particular in bin 3. The transfer does not seem to have an effect for the fuzzy C4.5 since the statistical errors remain close. However, the transfer failed for the FCGAM with increased statistical errors compared to the errors without corrections. Although the contamination percentage is lower, fewer events are retained, thus impairing the statistical error. The selection threshold is indeed quite high for the transferred and retrained FCGAM. We saw in chapter 7 that the domain adaptation worked better for π^0 production events, which constitute the training dataset, than for DVCS events that have a different topology.

To conclude on this topic, domain adaptation still requires further investigations so that it is more robust and reliable. Momentum corrections are great substitute in the meantime.

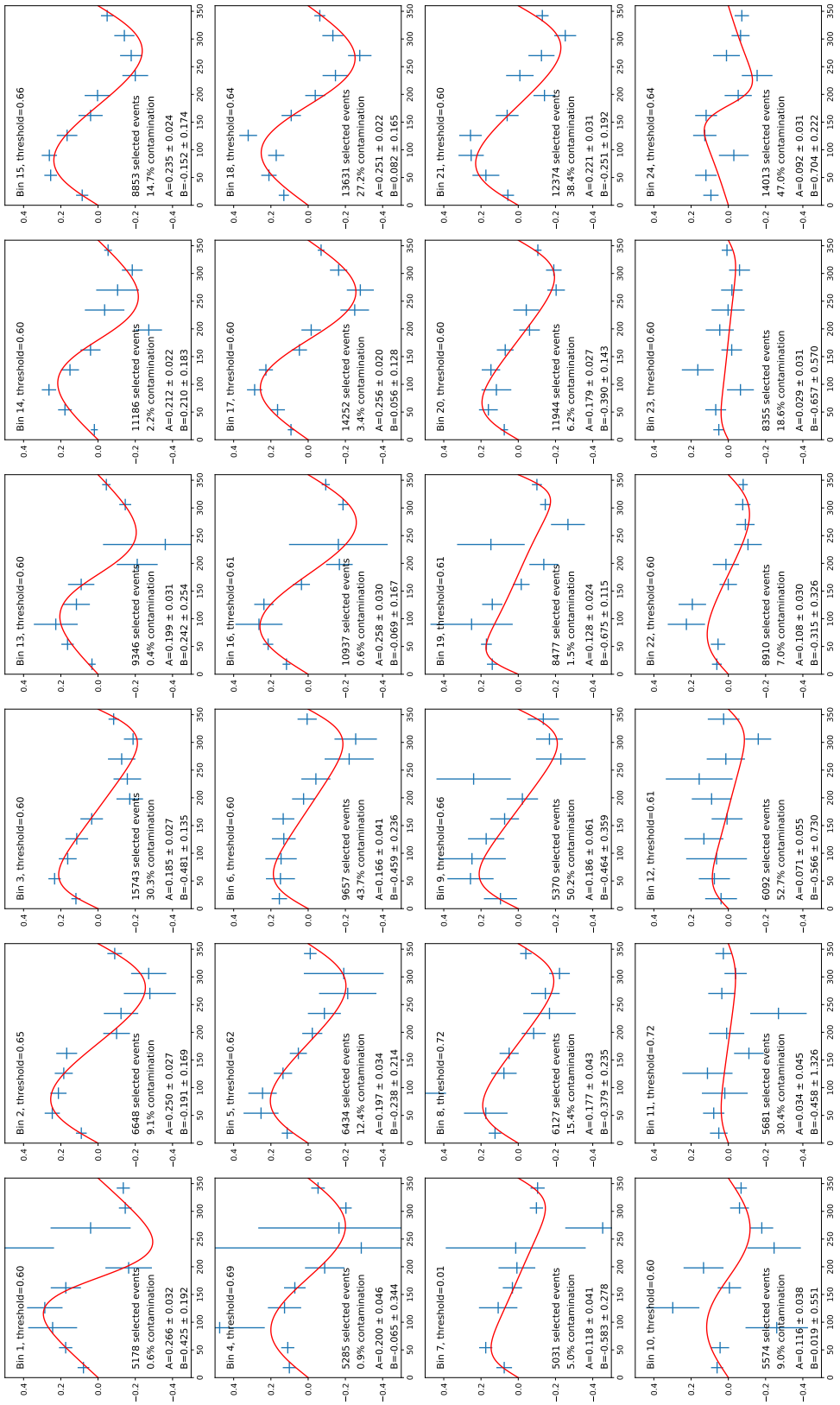


FIGURE 9.13: Asymmetries in each bin as function of the kinematic parameter ϕ, computed using the FCGAM.

9.4.2 Alternative methods for event selection

9.4.2.1 Cuts

Guillaume Christiaens from the University of Glasgow first determined a set of cuts on Monte Carlo simulated data from a base of five high-level variables. The thresholds on these variables were determined to minimize the statistical error on the asymmetry. Contrary to the approach based on machine learning models where only one threshold on the model's output must be determined, here a multidimensional cut must be found. The cuts determined on Monte Carlo simulated data (they were presented in chapter 8 for the interpretability survey) are then "transferred" to real data looking at the resolutions of the involved high-level variables. It is actually a renormalization strategy from the viewpoint of domain adaptation. The resulting cuts for real data are:

- the missing mass $e\gamma$ is in $\left[0.1\,\mathrm{GeV/c^2}, 1.7\,\mathrm{GeV/c^2}\right]$;
- angle $\left(\overrightarrow{p^{\gamma_1}}, \overrightarrow{e^{in}} - \overrightarrow{p^e} - \overrightarrow{p^p}\right) \leq 1.2°$ (photon cone angle);
- $\sqrt{\left(p_x^e + p_x^p + p_x^{\gamma_1}\right)^2 + \left(p_y^e + p_y^p + p_y^{\gamma_1}\right)^2} \leq 0.12\,\mathrm{GeV/c}$ (missing p_T);
- the squared missing mass $ep\gamma$ is in $\left[-0.04\,\mathrm{GeV^2/c^4}, 0.04\,\mathrm{GeV^2/c^4}\right]$;
- the missing energy $ep\gamma$ is in $[-0.5\,\mathrm{GeV}, 1.2\,\mathrm{GeV}]$.

Varying the values of the cuts on Monte Carlo simulated data permits to obtain efficiency against purity curves. However, it implies multivariate operations instead of only considering the model output for the proposed machine learning models.

As perspectives for this method, cuts could be differentiated depending on the kinematic bin. Similarly to the machine learning approach that we presented, this would permit to improve the statistical error on the asymmetry in each bin. However, customizing the cuts in each bin would mean a degraded interpretability in the machine learning viewpoint: indeed, the overall model would be more complex because divided in 24 cases. This is left to the judgment of the physicist in charge of the analysis.

9.4.2.2 Neural network

A study using neural networks to analyze CLAS12 data started being conducted by Marouen Baalouch from CEA LIST but has not been finalized. Marouen Baalouch compared several sets of features to use as input for the neural network, and concluded that using low-level features such as calorimeter responses is very beneficial to improve the classification performance. However, this raises a problem with respect to the π^0 subtraction since the decayed π^0 production events do not go through the complete event reconstruction. Instead, the two photons decayed from the π^0 simply undergo cuts on their energy and fiducial cuts to determine whether they are detected in the calorimeters. With the current analysis scheme, methods using low-level features are not exploitable.

We still picked up the designed network architectures and retrained them on Monte Carlo simulated data, with only the particles' three-momenta as input. This network was used for the interpretability survey in chapter 8.

Marouen Baalouch also studied DANN to handle the covariate shift between Monte Carlo simulation and real data [Baalouch et al., 2019]. As perspective for this work, a

DANN should be trained and tested using exclusive π^0 production as control process to learn domain-invariant features while performing DVCS event selection. The DANN should improve the classification performance of the neural network on real data, in the same way we performed domain adaptation to transfer our transparent models.

9.4.2.3 Results

In addition to comparing to the cuts and the neural network, we examine to which extent simplifying the FCGAM and FURIA to improve interpretability degrades the selection performances. We reduce the FCGAM to 5 terms and the fuzzy C4.5 to a maximal depth of 3. Table 9.2 compares the statistics (i.e. number of selected events) and contamination percentage in three representative bins: bin 3 comprises many contaminating events, bin 17 almost none, and bin 2 is an intermediate.

TABLE 9.2: Statistics and contamination percentage in three bins with several models.

	Bin 2		Bin 3		Bin 17	
	Stat.	Cont.	Stat.	Cont.	Stat.	Cont.
Cuts	5434	10.4%	11517	33.6%	12459	3.8%
Neural network	6287	**8.5%**	15450	30.9%	14456	3.5%
FURIA	**8148**	12.1%	**20211**	37.6%	**15649**	5.6%
FCGAM	6648	9.1%	15743	**30.3%**	14252	**3.4%**
FCGAM 5 terms	6828	9.7%	16618	33.2%	14007	**3.4%**
Fuzzy C4.5	5588	9.1%	15293	32.0%	13972	**3.4%**
Fuzzy C4.5 depth 3	5504	9.5%	14263	33.7%	11880	2.6%

FURIA is systematically obtaining the larger number of selected events, at the expense of a higher contamination level. The neural network is having the lower contamination level in bin 2, while it is the FCGAM in bin 3. Three models are tied in bin 17: the FCGAM, the same FCGAM limited to 5 terms, and the fuzzy C4.5. The complete FCGAM is the one retaining the most statistics. Overall, excluding FURIA, the cuts have the lowest statistics and the highest contamination in all bins. This proves the great interest of machine learning for a physics analysis: for instance, using the FCGAM permits to increase the statistics by 24% in average over these three bins, while reducing the contamination. This is equivalent to 24% additional beamtime. Finally, simplifying the FCGAM or the fuzzy C4.5 translates into only slight degradations of the results.

Table 9.3 compares the five main methods in a more complete way by summarizing the percentage of bins in which the model in line obtains a better asymmetry error than the model in column.

FURIA is the only model surpassing all the others. However, these results should be taken with very high precaution: indeed, FURIA is keeping much more events than the other models, including 13% of identified η production events (see 9.1.3). Therefore, its performance must be read with caution since there is a high associated systematic error, linked to other background processes that were not properly removed. FURIA is not applicable as is for this particular analysis, unless proper weights are set during training, which was not done in this study. On the contrary, all methods outperform the cuts. This proves again the interest of machine learning to enhance the analysis.

	Cuts	Neural network	FURIA	Fuzzy C4.5	FCGAM
Cuts	/	8%	17%	29%	12%
Neural network	88%	/	21%	54%	38%
FURIA	83%	79%	/	71%	63%
Fuzzy C4.5	71%	38%	29%	/	21%
FCGAM	88%	58%	37%	71%	/

TABLE 9.3: Percentage of winning bins of the models in lines, against the models in columns, in terms of asymmetry error.

The FCGAM is better than all other models except FURIA. This includes the neural network, which is surprising considering the ROC curves presented at the beginning of this chapter (Figure 9.3). However, considering per bin comparisons, the FCGAM is more relevant.

9.5 Conclusion and perspectives

This chapter concludes the thesis with the actual DVCS data analysis at CLAS12. We applied the previously developed transparent models with automatically built features on preselected CLAS12 data. Using the π^0 subtraction procedure to estimate π^0 contamination, we were able to validate the proper functioning of the different models. The selection threshold of the models has been fine-tuned in each kinematic bin to minimize the asymmetry error, the DVCS asymmetry being the final observable we want to measure. We produced asymmetry plots with several analysis techniques (cuts, transparent models, neural network), which are all consistent with each other and follow the expected shape.

The proposed FCGAM turns out to be a very satisfying model: it was rated as the preferred model in terms of interpretability by the physicists (chapter 8) and obtains similar or better performances than the neural network for the data analysis. Overall, all of the proposed transparent models perform honourably and significantly improve the usual physics cuts for the data analysis. Compared to the cuts, the FCGAM increases the statistics by 24%. It is equivalent to having 24% of free additional beam time. These results constitute an excellent proof of the interest of our models.

To pursue this work, more attention should be paid to other background processes. We saw that η production events can be bothersome for a few models. An option would be to subtract this process following the same guidelines as for π^0 subtraction. Another potentially contaminating process is the Δ-VCS: the final state consists of an electron, a photon and a Δ baryon, an excited state of the proton. This Δ particle decays either into a π^0 and a proton, or into a π^+ and a neutron. In the first case, the event can be confounded with a DVCS. Actually, a proper way of dealing with additional background is to include more classes during model induction. Here, η production events were neither classified as π^0 events nor as DVCS events, which is logical since these events were not included in the training dataset. Adding more background classes or simply merging all of them into a background class as opposed to the signal class for DVCS would permit to eliminate such processes more surely. In the first place, we should produce simulations of these processes for training. However, this is not necessarily a simple task since the cross-sections may be poorly known, as this is the case for Δ-VCS.

Conclusion of part III

This part was dedicated to ensure the transparent models proposed in part II are applicable to real CLAS12 data.

Chapter 7 started by tackling the distribution shifts between Monte Carlo simulation and CLAS12 data. Due to uncertainties in the experimental setup and theoretical models, the agreement between experimental and simulated data is degraded. We proposed a domain adaptation technique using generative neural networks to map the simulation distributions to the real data distributions. However, since the control process used to learn the mapping covers only a limited region of the phase space, this mapping is not entirely generalizable. This work still demonstrates the feasibility of such an approach thanks to promising results on flat generated data and establishes guidelines for further research.

Chapter 8 discussed the interpretability of the proposed models, from a subjective point of view. Indeed, the proposed transparent models and the features they include should be interpretable to the physicist users. Therefore, we designed a survey aiming at evaluating the perception of physicists of our built features and models. We believe such a study is of high interest for both the interpretable machine learning field and the experimental physics domain. The results of the survey validated the understandability of the features built with constraints, and the interpretability notably of the GAM and of FURIA to a lesser extent. Globally, the respondents particularly appreciated the proposed GAM model.

Finally, we performed in chapter 9 the final physics analysis of CLAS12 data, using the proposed transparent models. The advantage of GAM and decision trees over FURIA are their flexibility in the decision threshold to optimize the asymmetry statistical error. We validated the functioning of the GAM and fuzzy C4.5 thanks to the observation of the models themselves applied to CLAS12 data. Finally, we extracted DVCS asymmetries using these models, which are the final observables we wanted to extract to help the investigations of the proton structure. Compared to the physicists' cuts, significant improvements of the results have been obtained: more events are retrieved with lower π^0 contamination. Concretely, 24% more statistics are retained for an average decrease of 11% of the contamination. Selecting more events is particularly crucial for bins with low statistics, notably at high Q^2 values. Finally, the results are comparable between the neural network and the transparent models thanks to feature construction, but the neural network cannot be validated in the same way as for the GAM and fuzzy C4.5. To complete the analysis, a detailed study of the systematic errors must be conducted. These errors are common to all analysis methods.

Conclusion

This thesis has the objective to propose and adapt transparent machine learning models for the analysis of DVCS data at CLAS12. Indeed, event selection is a crucial step in a physics analysis that can be optimized, thanks to computing tools, more easily than detector performances that have already received large investments. The transparency of the models is necessary to guarantee their validation for application to real data and peer-reviewing before publication. To ensure the performances of such models for event selection, this thesis aims at upgrading their classification performances while guaranteeing their interpretability.

Thus, part I started by presenting the context of the studies and our positioning. First, the physics theoretical and experimental background was introduced, to understand the physics challenges. Then, we proposed a review of the state of the art in interpretable machine learning. Evaluation protocols of the interpretability of machine learning models were notably discussed. In particular, we concluded that transparent machine learning models often suffer from a gap in classification performance compared to so-called "black-box" models. We initiated the analysis workflow by producing simulated data for model induction. In particular, flat generation must be privileged to avoid biasing the models towards the cross-sections distributions. We implemented a few transparent models as baselines.

Part II dived into the enhancement of the performances and interpretability of transparent machine learning models. We developed a new constrained feature construction algorithm that is able to build consistent and understandable features for HEP applications. Thanks to grammars and transition matrices, the built features follow expert knowledge about physics laws. This algorithm was successfully applied, first as a prior method using various fitness functions, and secondly as an embedded method in tree-based models, FURIA and GAM. For the latter, we also defined bitonicity as a new constraint to enforce so that the model has a bitonic output with respect to any bitonic input feature. This contribution permits to strengthen the interpretability of the model, especially for bitonic features, found in abundance in HEP. The methodologies developed in this part can actually be applied to many other experiments in HEP: we gave a few examples with the Higgs, $\tau \rightarrow 3\mu$ and MAGIC datasets since they are publicly available, but other physics experiments could be considered. More generally, any data analysis problem involving input variables having dimensions and units could benefit from our approach.

Part III focused on the application of the enhanced models to real CLAS12 data. Although this part is more specific to CLAS12, the analysis workflow can be extracted and adapted to other HEP experiments. First, we must take care of the distribution shifts between simulation and real data that can be due to multiple factors including uncertainties on the experimental setup or on physics theory. We proposed a domain adaptation technique based on generative networks that was experimentally proven to work on flat generated data. Then, to be applicable to real data, the interpretability

of the models as perceived by our physicists colleagues must be assessed. To this end, we performed a complete interpretability evaluation based on a survey to which 31 physicists responded. Such a study is of high interest both for the interpretable machine learning community and for the physicists. The results confirmed and validated the interest of the respondents for the proposed models, to be used in a physics analysis. Finally, we successfully conducted the DVCS analysis using the proposed models: the removal of other background processes was ensured by the choice of a minimal selection threshold. Then, π^0 subtraction permitted to determine the optimal selection threshold to minimize the statistical error on the DVCS asymmetry, our goal observable. Observing the distributions of the main variables used by our models permitted to validate their consistency. We also compared the proposed transparent models to two alternatives for event selection: standard cuts on the one hand, and a neural network on the other hand. While the proposed transparent models largely outperform the cuts, the GAM notably obtains similar or better results (i.e. statistical error) than the neural network. This is a great conclusion for our approach, since we are able to combine both interpretability and classification performance with the GAM.

In the introduction of this thesis, we stated that a side objective of using transparent machine learning, apart from validating them, is to retro-engineer the induced models to maybe learn from them and improve the standard analysis. In this thesis, we discovered numerous discriminative high-level variables to be used to separate signal from background in data. For DVCS event selection, it turned out that using variables to identify π^0 production events is very beneficial to eliminate them. Notably, variables comparing two photons, if any, have been found to be very discriminative. Regarding DVCS-related variables, the sum of the z momenta of the output particles is also a very frequent variable found by the feature construction algorithm, including the unconstrained ones that have a larger search space available. Besides, the cuts themselves could be optimized thanks to machine learning: if we wanted to keep the same structure, i.e. a series of cuts on chosen variables, we could perform successive feature constructions and selection thresholds optimizations using for instance an information metric. We could also fuzzify these thresholds to take advantage of the flexibility fuzzy logic allows. To go further, GAM with few terms permit to boost the classification performances, thanks to the smooth shape functions.

Many opportunities can be imagined for future work. The feature construction proposed in part II is applicable to many other experiments in HEP, and is the core of the proposed transparent models for event selection. With provided samples of DVCS, π^0 production events, η production events or others, it is straightforward to apply the proposed algorithms. Notably, it could be of high interest to apply the feature construction algorithm to COMPASS or the future EIC data for instance, and study the differences with CLAS12 discriminative features. Of course, given the geometry of these experiments, the most interesting features will not be the same. The beam axis was favored in CLAS12 since it is a fixed target experiment, while at EIC the transverse plane will probably be more interesting given its geometry. Outside of the DVCS analysis, other experiments such as ATLAS (Higgs dataset) could of course benefit as well from this technique. However, since the challenges differ, more adaptations are probably needed.

Domain adaptation is of primary importance to ensure the proper functioning of the induced models on real data. We started the research in chapter 7 and identified the most promising technique being generative neural networks. We identified the

bottleneck of our current approach, which is our training set: consisting of exclusive π^0 production events, it does not cover the entire phase space. To further deepen this work and design a robust mapping between simulation and real data, a more representative dataset needs to be built. Selecting DVCS events with very strict cuts and performing π^0 subtraction should provide us with a dataset covering a larger part of the phase space. Directly domain-invariant models constitute an alternative, but fewer initiatives exist in the literature apart from domain-invariant neural networks. An idea would be to introduce domain-invariant metrics in the feature construction process: the built features would therefore be robust to the distributions shift.

Finally, the analysis needs to be perfected: the π^0 subtraction process is still ongoing work. Besides, improvements of the reconstruction are also in progress. In addition, a complete study of the systematic errors must be ultimately conducted before publication of the analysis results. They notably involve the acceptance of the detectors, the error on the estimation of the π^0 contamination fraction, the uncertainty on the beam polarization fraction, considerations of radiative corrections, etc. These errors are common to all analysis techniques and will not alter the comparative results presented in chapter 9. To further improve the statistics retained by the different models, we should consider including more background processes during the training process. Notably, η production and Δ-VCS are possible contaminations to the DVCS. In addition, all the methods we proposed in this thesis for event selection are compatible with missing values and are applicable in particular when the proton is missing. In chapter 9, we only considered full events with an electron, a proton and at least a photon, while we expect that the number of events could be approximately doubled by considering events without proton.

The works accomplished in this thesis led to a number of publications and talks, listed on the next page.

Publications and talks

Publications

International conferences

N. Cherrier, J. P. Poli, M. Defurne and F. Sabatié, "Consistent Feature Construction with Constrained Genetic Programming for Experimental Physics", *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1650-1658, IEEE, 2019.

N. Cherrier, J. P. Poli, M. Defurne and F. Sabatié, "Embedded Feature Construction in Fuzzy Decision Tree Induction for High Energy Physics Classification", *2020 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 615-622, IEEE, 2020.

N. Cherrier, M. Mayo, J. P. Poli, M. Defurne and F. Sabatié, "Interpretable Machine Learning with Bitonic Generalized Additive Models and Automatic Feature Construction", *International Conference on Discovery Science*, pages 386-402, Springer, Cham, 2020.

Workshops

M. Baalouch, M. Defurne, J. P. Poli, F. Sabatié and N. Cherrier, "Sim-to-Real Domain Adaptation For High Energy Physics", *Machine Learning and the Physical Sciences workshop at the 33rd Conference on Neural Information Processing Systems*, 2019.

N. Cherrier, M. Defurne, J. P. Poli and F. Sabatié, "Embedded Constrained Feature Construction for High-Energy Physics Data Classification", *Machine Learning and the Physical Sciences workshop at the 33rd Conference on Neural Information Processing Systems*, 2019.

Talks

"Construction de features interprétables pour la classification en physique expérimentale", *GT explicabilité*, 2018.

"Interpretable machine learning techniques for CLAS12 data analysis", *Artificial intelligence and physics conference (PhysAI)*, 2019.

"Interpretable ML for CLAS12 data analysis: adaptation of Generalized Additive Models", *IN2P3/IRFU Machine Learning workshop*, 2020.

"Interpretable machine learning for CLAS12 data analysis", *InTheArt*, 2020.

"Event classification with ML at CLAS12", *A.I. for Nuclear Physics workshop*, 2020.

Bibliography

A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

A. Accardi, J. Albacete, M. Anselmino, N. Armesto, E. Aschenauer, A. Bacchetta, D. Boer, W. Brooks, T. Burton, N.-B. Chang, et al. Electron-ion collider: The next QCD frontier. *The European Physical Journal A*, 52(9):268, 2016.

C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kegl, and D. Rousseau. Learning to discover: the Higgs boson machine learning challenge. 9, 2014. URL http://higgsml.lal.in2p3.fr/documentation.

R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pages 487–499, 1994.

S. Ahmed, M. Zhang, and L. Peng. A new GP-based wrapper feature construction approach to classification and biomarker identification. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2756–2763. IEEE, 2014a.

S. Ahmed, M. Zhang, L. Peng, and B. Xue. Multiple feature construction for effective biomarker identification and classification using genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 249–256, 2014b.

S. Ahmed, M. Zhang, L. Peng, and B. Xue. A multi-objective genetic programming biomarker detection approach in mass spectrometry data. In *European Conference on the Applications of Evolutionary Computation*, pages 106–122. Springer, 2016.

Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco, and Y. Li. Simulation of electron-proton scattering events by a feature-augmented and transformed generative adversarial network (FAT-GAN). *arXiv preprint arXiv:2001.11103*, 2020.

J. Alcala-Fdez, R. Alcala, and F. Herrera. A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy systems*, 19(5):857–872, 2011.

H. Allahyari and N. Lavesson. User-oriented assessment of classification model understandability. In *11th scandinavian conference on Artificial intelligence*. IOS Press, 2011.

J. M. Alonso and A. Bugarín. ExpliClas: automatic generation of explanations in natural language for WEKA classifiers. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2019.

A. Altay and D. Cinar. Fuzzy decision trees. In *Fuzzy Statistical Decision-Making*, pages 221–261. Springer, 2016.

D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

R. Ambrosino, B. G. Buchanan, G. F. Cooper, and M. J. Fine. The use of misclassification costs to learn rule-based decision support models for cost-effective hospital admission strategies. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 304. American Medical Informatics Association, 1995.

M. Antonelli, P. Ducange, F. Marcelloni, and A. Segatori. A novel associative classification model based on a fuzzy frequent pattern mining algorithm. *Expert Systems with Applications*, 42(4):2086–2097, 2015.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

L. Arras, G. Montavon, K.-R. Müller, and W. Samek. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.

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

M. W. Aslam, Z. Zhu, and A. K. Nandi. Feature generation using genetic programming with comparative partner selection for diabetes classification. *Expert Systems with Applications*, 40(13):5402–5412, 2013.

I. Baaj and J.-P. Poli. Natural language generation of explanations of fuzzy inference decisions. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2019.

M. Baalouch, M. Defurne, J.-P. Poli, and N. Cherrier. Sim-to-real domain adaptation for high energy physics. *arXiv preprint arXiv:1912.08001*, 2019.

P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic programming: An introduction on the automatic evolution of computer programs and its applications*. 1998.

T. Barthelemy. On the unimodality of METRIC approximation subject to normally distributed demands. 2015.

A. Barua, L. S. Mudunuri, and O. Kosheleva. Why trapezoidal and triangular membership functions work so well: Towards a theoretical explanation. 2013.

S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Y. Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8):1798–1828, 2013.

S. Bethke. The 2009 world average of α s. *The European Physical Journal C*, 64(4): 689–703, 2009.

B. Bhushan Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty. DeepJDOT: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 447–463, 2018.

C. Borgelt, J. Gebhardt, and R. Kruse. Concepts for probabilistic and possibilistic induction of decision trees on real world data. *Proc. of the EUFIT*, 96:1556–1560, 1996.

B. Bouchon-Meunier and C. Marsala. Learning fuzzy decision rules. In *Fuzzy Sets in Approximate Reasoning and Information Systems*, pages 279–304. Springer, 1999.

B. Bouchon-Meunier and C. Marsala. Measures of discrimination for the construction of fuzzy decision trees. In *International Conference on Fuzzy Information Processing*, pages 709–714, 2003.

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

P. Bujnowski, E. Szmidt, and J. Kacprzyk. An approach to intuitionistic fuzzy decision trees. In *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)*. Atlantis Press, 2015.

V. Burkert, L. Elouadrhiri, K. Adhikari, S. Adhikari, M. Amaryan, D. Anderson, G. Angelini, M. Antonioli, H. Atac, S. Aune, et al. The CLAS12 Spectrometer at Jefferson Laboratory. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 959: 163419, 2020.

R. Calabrese et al. Estimating bank loans loss given default by generalized additive models. *UCD Geary Institute Discussion Paper Series, WP2012/24*, 2012.

A. Campagner, F. Cabitza, and D. Ciucci. Three-way decision for handling uncertainty in machine learning: A narrative review. In *International Joint Conference on Rough Sets*, pages 137–152. Springer, 2020.

A. Cano, S. Ventura, and K. J. Cios. Multi-objective genetic programming for feature extraction and data visualization. *Soft Computing*, 21(8):2069–2089, 2017.

G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.

R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission.

In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

N. Chakravarti. Isotonic median regression: a linear programming approach. *Mathematics of operations research*, 14(2):303–308, 1989.

B. Chandra and P. P. Varghese. Fuzzifying Gini index based decision trees. *Expert Systems with Applications*, 36(4):8549–8559, 2009.

G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

G. Charles. *Development of Micromegas detectors for the CLAS12 experiment at Jefferson Laboratory.* PhD thesis, Université Paris Sud - Paris XI, 2013.

M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

Q. Chen, M. Zhang, and B. Xue. Genetic programming with embedded feature construction for high-dimensional symbolic regression. In *Intelligent and Evolutionary Systems*, pages 87–102. Springer, 2017.

Q. Chen, Y. Liu, Z. Wang, I. Wassell, and K. Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

S. Chopra, S. Balakrishnan, and R. Gopalan. DLID: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*, volume 2, 2013.

N. Chouika. *Generalized Parton Distributions and their covariant extension: towards nucleon tomography.* PhD thesis, Université Paris-Saclay, 2018.

G. Christiaens. *Beam-spin asymmetry of deeply virtual Compton scattering off the proton at 10.6 GeV with CLAS12 at Jefferson Laboratory.* PhD thesis, University of Glasgow, 2021.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine learning*, 3:261–283, 1989.

I. Cloete and J. van Zyl. Fuzzy rule induction in a set covering framework. *IEEE Transactions on Fuzzy Systems*, 14(1):93–110, 2006.

W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Elsevier, 1995.

G. Cormode and A. McGregor. Approximation algorithms for clustering uncertain data. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 191–200, 2008.

P. Cortez and M. J. Embrechts. Opening black box data mining models using sensitivity analysis. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 341–348. IEEE, 2011.

P. Cortez and M. J. Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1–17, 2013.

N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy. Joint distribution optimal transportation for domain adaptation, 2017.

K. Cpałka. *Design of Interpretable Fuzzy Systems*, volume 684. Springer, 2017.

L. Crochepierre, L. Boudjeloud-Assala, and V. Barbesant. Interpretable dimensionally-consistent feature extraction from electrical network sensors. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML/PKDD'20*, 2020.

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

W. M. Czarnecki and I. T. Podolak. Machine learning with known input data uncertainty measure. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 379–388. Springer, 2013.

Y. Dai, B. Xue, and M. Zhang. New representations in PSO for feature construction in classification. In *European Conference on the Applications of Evolutionary Computation*, pages 476–488. Springer, 2014.

L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman. Jet-images—deep learning edition. *Journal of High Energy Physics*, 2016(7):69, 2016.

L. de Oliveira, M. Paganini, and B. Nachman. Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1):4, 2017.

M. Defurne. *Photon and π^0 electroproduction at Jefferson Laboratory-Hall A*. PhD thesis, Université Paris Sud - Paris XI, 2015.

M. J. Del Jesus, F. Hoffmann, L. J. Navascués, and L. Sánchez. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE Transactions on Fuzzy Systems*, 12(3):296–308, 2004.

C. Denis and F. Varenne. Interprétabilité et explicabilité pour l'apprentissage machine: entre modèles descriptifs, modèles prédictifs et modèles causaux. une nécessaire clarification épistémologique. 2019.

L. M. Dery, B. Nachman, F. Rubbo, and A. Schwartzman. Weakly supervised classification in high energy physics. *Journal of High Energy Physics*, 2017(5):1–11, 2017.

Y. Dong, H. Su, J. Zhu, and B. Zhang. Improving interpretability of deep neural networks with semantic information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4306–4314, 2017.

O. Dor and Y. Reich. Strengthening learning algorithms by feature discovery. *Information Sciences*, 189:176–190, 2012.

M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, pages 1470–1477. IEEE, 1999.

M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

F. K. Došilović, M. Brčić, and N. Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

D. Drechsel, O. Hanstein, S. Kamalov, and L. Tiator. A unitary isobar model for pion photo-and electroproduction on the proton up to 1 GeV. *Nuclear Physics A*, 645 (1):145–174, 1999.

K. Drozdz and H. Kwasnicka. Feature set reduction by evolutionary selection and construction. In *KES International Symposium on Agent and Multi-agent Systems: Technologies and Applications*, pages 140–149. Springer, 2010.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

D. Dubois and H. Prade. Possibility theory. In *Computational Complexity: Theory, Techniques, and Applications*, pages 2240–2252. Springer, 2012.

D. Dubois, E. Hüllermeier, and H. Prade. A systematic approach to the assessment of fuzzy association rules. *Data Mining and Knowledge Discovery*, 13(2):167–192, 2006.

G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

A. Ekárt and A. Márkus. Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*, 17(3):205, 2003.

A. Elola, J. Del Ser, M. N. Bilbao, C. Perfecto, E. Alexandre, and S. Salcedo-Sanz. Hybridizing Cartesian genetic programming and harmony search for adaptive feature construction in supervised learning problems. *Applied Soft Computing*, 52: 760–770, 2017.

T. Elomaa. In defense of C4.5: Notes on learning one-level decision trees. In *Machine Learning Proceedings 1994*, pages 62–69. Elsevier, 1994.

F. Emmert-Streib, O. Yli-Harja, and M. Dehmer. Explainable artificial intelligence and machine learning: A reality rooted perspective. *arXiv preprint arXiv:2001.09464*, 2020.

M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt. Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks. *Computing and Software for Big Science*, 2(1):4, 2018.

M. Erdmann, J. Glombitza, and T. Quast. Precise simulation of electromagnetic calorimeter showers using a Wasserstein generative adversarial network. *Computing and Software for Big Science*, 3(1):4, 2019.

European Parliament and Council of European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.

R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

W. Fan, E. Zhong, J. Peng, O. Verscheure, K. Zhang, J. Ren, R. Yan, and Q. Yang. Generalized and heuristic-free feature construction for improved accuracy. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 629–640. SIAM, 2010.

A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. A brief review of domain adaptation. *arXiv preprint arXiv:2010.03978*, 2020.

M. M. Fard, K. Canini, A. Cotter, J. Pfeifer, and M. Gupta. Fast and flexible monotonic functions with ensembles of lattices. In *Advances in Neural Information Processing Systems*, pages 2919–2927, 2016a.

M. M. Fard, K. Canini, A. Cotter, J. Pfeifer, and M. Gupta. Fast and flexible monotonic functions with ensembles of lattices. In *Advances in neural information processing systems*, pages 2919–2927, 2016b.

T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

Federal government of the United States. Equal Credit Opportunity Act (Regulation B of the Code of Federal Regulations), Title 12, Chapter X, Part 1002, §1002.9, 1974.

D. E. Ferguson. Fibonaccian searching. *Communications of the ACM*, 3(12):648, 1960.

A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera. Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. *IEEE Transactions on Evolutionary Computation*, 14(6):913–941, 2010.

C. S. Fertig, A. A. Freitas, L. V. Arruda, and C. Kaestner. A fuzzy beam-search rule induction algorithm. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 341–347. Springer, 1999.

R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188, 1936.

R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

E. Frank, M. A. Hall, and I. H. Witten. *The WEKA workbench*. Morgan Kaufmann, 2016.

A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, 2014.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2: 721–747, 2002.

J. Fürnkranz and T. Kliegr. A brief overview of rule learning. In *International symposium on rules and rule markup languages for the semantic web*, pages 54–69. Springer, 2015.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

D. García, A. González, and R. Pérez. A feature construction approach for genetic iterative rule learning algorithm. *Journal of Computer and System Sciences*, 80(1): 101–117, 2014.

D. García, D. Stavrakoudis, A. González, R. Pérez, and J. B. Theocharis. A fuzzy rule-based feature construction approach applied to remotely sensed imagery. In *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)*. Atlantis Press, 2015.

D. Gavrilis and I. Tsoulos. Classification of fetal heart rate using grammatical evolution. In *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.*, pages 425–429. IEEE, 2005.

D. Gavrilis, I. G. Tsoulos, and E. Dermatas. Selecting and constructing features using grammatical evolution. *Pattern Recognition Letters*, 29(9):1358–1365, 2008.

H. Gegier and E. Marsden. On a diffuse reflection of the α-particles. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 82(557):495–500, 1909.

M. Gell-Mann. A schematic model of baryons and mesons. *Physics Letters*, 8(3): 214–215, 1964.

M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

A. Ghosh et al. Deep generative models for fast shower simulation in ATLAS. In *Journal of Physics: Conference Series*, volume 1525, page 012077. IOP Publishing, 2020.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An approach to evaluating interpretability of machine learning. *arXiv preprint arXiv:1806.00069*, 2018.

V. V. Gligorov and M. Williams. Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree. *Journal of Instrumentation*, 8(02):P02013, 2013.

X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

K. Goeke, M. V. Polyakov, and M. Vanderhaeghen. Hard exclusive reactions and the structure of hadrons. *arXiv preprint hep-ph/0106012*, 2001.

A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

S. Goloskokov and P. Kroll. Transversity in hard exclusive electroproduction of pseudoscalar mesons. *The European Physical Journal A*, 47(9):112, 2011.

B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012.

A. González and R. Pérez. SLAVE: A genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191, 1999.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, pages 999–1006. IEEE, 2011.

D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13(2-3):229–257, 1993.

A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4): 5, 2009.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

C. Gu and G. Wahba. Minimizing GCV/GML scores with multiple smoothing parameters via the Newton method. *SIAM Journal on Scientific and Statistical Computing*, 12(2):383–398, 1991.

M. Guidal, M. Polyakov, A. Radyushkin, and M. Vanderhaeghen. Nucleon form factors from generalized parton distributions. *Physical Review D*, 72(5):054013, 2005.

M. Guidal, H. Moutarde, and M. Vanderhaeghen. Generalized parton distributions in the valence region from deeply virtual compton scattering. *Reports on Progress in Physics*, 76(6):066202, 2013.

R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018a.

R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018b.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

G. F. Gunawan, S. C. Gosaria, and A. Z. Arifin. Grammatical evolution for feature extraction in local thresholding problem. *Jurnal Ilmu Komputer dan Informasi*, 5 (2):106–111, 2012.

D. Gunning. Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017.

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)*, 35(1):89–99, 2005.

L. Guo, D. Rivero, J. Dorado, C. R. Munteanu, and A. Pazos. Automatic feature extraction using genetic programming: An application to epileptic EEG classification. *Expert Systems with Applications*, 38(8):10425–10436, 2011.

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

M. A. Hall. Correlation-based feature selection for machine learning. 1999.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2000.

E. Hart, K. Sim, B. Gardiner, and K. Kamimura. A hybrid method for feature construction and selection to improve wind-damage prediction in the forestry sector. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1121–1128, 2017.

D. Y. Harvey and M. D. Todd. Automated Feature Design for Numeric Sequence Classification by Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 19(4):474–489, 2014.

T. J. Hastie and R. J. Tibshirani. Generalized additive models. *Statistical Science*, 1 (3):297–310, 1986.

T. D. Haynes, D. A. Schoenefeld, and R. L. Wainwright. Type inheritance in strongly typed genetic programming. *Advances in genetic programming*, 2(2):359–376, 1996.

L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

J. Heo, S. Joo, and T. Moon. Fooling neural network interpretations via adversarial model manipulation. In *Advances in Neural Information Processing Systems*, pages 2921–2932, 2019.

F. Herrera. Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27–46, 2008.

N. Holden and A. A. Freitas. A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *Journal of Artificial evolution and Applications*, 2008, 2008.

T.-P. Hong and Y.-C. Lee. An overview of mining fuzzy association rules. In *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, pages 397–410. Springer, 2008.

G. Huang, S. Song, C. Wu, and K. You. Robust support vector regression for uncertain input and output data. *IEEE Transactions on Neural Networks and Learning Systems*, 23(11):1690–1700, 2012.

J. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, 2009.

J. C. Hühn and E. Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions on Fuzzy Systems*, 17(1):138–149, 2008.

D. Huk Park, L. Anne Hendricks, Z. Akata, A. Rohrbach, B. Schiele, T. Darrell, and M. Rohrbach. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8779–8788, 2018.

E. Hüllermeier. Does machine learning need fuzzy logic? *Fuzzy Sets and Systems*, 281:292–299, 2015.

I. Icke and A. Rosenberg. Multi-objective genetic programming for visual analytics. In *European Conference on Genetic Programming*, pages 322–334. Springer, 2011.

H. Ishibuchi, T. Murata, and I. Türkşen. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy sets and systems*, 89(2):135–150, 1997.

C. Z. Janikow. Fuzzy decision trees: issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(1):1–14, 1998.

C. Z. Janikow. FID4.1: an overview. In *IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS'04.*, volume 2, pages 877–881. IEEE, 2004.

F. Janssen and J. Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, 78(3):343–379, 2010.

I. Jenhani, N. B. Amor, and Z. Elouedi. Decision trees as possibilistic classifiers. *International journal of approximate reasoning*, 48(3):784–807, 2008.

X. Ji. Gauge-invariant decomposition of nucleon spin. *Physical Review Letters*, 78(4): 610, 1997.

U. Johansson, R. König, and L. Niklasson. The truth is in there-rule extraction from opaque models using genetic programming. In *FLAIRS Conference*, pages 658–663. Miami Beach, FL, 2004a.

U. Johansson, L. Niklasson, and R. König. Accuracy vs. comprehensibility in data mining models. In *Proceedings of the seventh international conference on information fusion*, volume 1, pages 295–300, 2004b.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 1994.

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag New York, Inc., 2 edition, 2002.

J. M. Kanter and K. Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

A. Kaul, S. Maheshwary, and V. Pudi. Autolearn — automated feature generation and selection. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 217–226. IEEE, 2017.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, page 3149–3157, 2017.

M. Keijzer and V. Babovic. Dimensionally aware genetic programming. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1069–1076, 1999.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

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)*, pages 1304–1307. IEEE, 2016.

U. Khurana, H. Samulowitz, and D. Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems*, pages 2280–2288, 2016.

P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (un)reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

J. Klaas. *Machine learning for finance: principles and practice for financial insiders*. Packt Publishing Ltd, 2019.

R. Kohavi, G. H. John, et al. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

C. Köllmann, B. Bornkamp, and K. Ickstadt. Unimodal regression using Bernstein–Schoenberg splines and penalties. *Biometrics*, 70(4):783–793, 2014.

R. Konig, U. Johansson, and L. Niklasson. G-REX: A versatile framework for evolutionary data mining. In *2008 IEEE International Conference on Data Mining Workshops*, pages 971–974. IEEE, 2008.

W. Kotłowski and R. Słowiński. Rule learning with monotonicity constraints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 537–544, 2009.

S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39 (4):261–283, 2013.

W. M. Kouw and M. Loog. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5686–5697, 2016.

K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3 (4):329–343, 2002.

K. Krawiec and L. Włodarski. Coevolutionary feature construction for transformation of representation of machine learners. In *Intelligent Information Processing and Web Mining*, pages 139–150. Springer, 2004.

H.-P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 672–677, 2005.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

I. Kuralenok, V. Ershov, and I. Labutin. MonoForest framework for tree ensemble analysis. In *Advances in neural information processing systems*, pages 13780–13789, 2019.

I. Lage, E. Chen, J. He, M. Narayanan, B. Kim, S. Gershman, and F. Doshi-Velez. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*, 2019.

H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

H. T. Lam, J.-M. Thiebaut, M. Sinn, B. Chen, T. Mai, and O. Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294*, 2019.

H. Laurent and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information processing letters*, 5(1):15–17, 1976.

N. Lavrač. Selected techniques for data mining in medicine. *Artificial intelligence in medicine*, 16(1):3–23, 1999.

B. Letham, C. Rudin, T. H. McCormick, D. Madigan, et al. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

LHCb collaboration. Search for the lepton flavour violating decay $\tau\text{-}\rightarrow \mu\text{-}\ \mu+\ \mu\text{-}$. *Journal of High Energy Physics*, 2015(2):121, 2015.

C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in neural information processing systems*, pages 2203–2213, 2017.

W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings 2001 IEEE international conference on data mining*, pages 369–376. IEEE, 2001.

Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

T. Likhomanenko, P. Ilten, E. Khairullin, A. Rogozhnikov, A. Ustyuzhanin, and M. Williams. LHCb topological trigger reoptimization. volume 664, page 082025, 2015.

S. H. Lim, L.-L. Wang, and G. DeJong. Explanation-based feature construction. In *Proceedings of the 20th international joint conference on Artifical intelligence*, volume 7, pages 931–936, 2007.

Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

B. Liu, W. Hsu, Y. Ma, et al. Integrating classification and association rule mining. In *ACM International Conference on Knowledge Discovery and Data Mining*, volume 98, pages 80–86, 1998.

T. Lombrozo. Simplicity and probability in causal explanation. *Cognitive psychology*, 55(3):232–257, 2007.

M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013.

M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

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

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

F. Maes, P. Geurts, and L. Wehenkel. Embedding Monte Carlo search of features in tree-based ensemble methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 191–206. Springer, 2012.

A. Maevskiy, D. Derkach, N. Kazeev, A. Ustyuzhanin, M. Artemev, L. Anderlini, L. Collaboration, et al. Fast data-driven simulation of Cherenkov detectors using generative adversarial networks. In *Journal of Physics: Conference Series*, volume 1525. IOP Publishing, 2020.

A. Mahanipour and H. Nezamabadi-pour. Improved PSO-based feature construction algorithm using feature selection methods. In *2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 1–5. IEEE, 2017.

A. Mahanipour and H. Nezamabadi-pour. A multiple feature construction method based on gravitational search algorithm. *Expert Systems with Applications*, 127: 199–209, 2019.

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

P. E. Maher and D. S. Clair. Uncertain reasoning in an ID3 machine learning framework. In *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*, pages 7–12. IEEE, 1993.

S. Markovitch and D. Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49(1):59–98, 2002.

C. Marsala. Gradual fuzzy decision trees to help medical diagnosis. In *2012 IEEE International Conference on Fuzzy Systems*, pages 1–6. IEEE, 2012.

C. Marsala and D. Petturiti. Monotone classification with decision trees. In *EUSFLAT Conf.*, 2013.

D. Martens, B. Baesens, and T. Fawcett. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82(1):1–42, 2011.

J. A. Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant. Particle generative adversarial networks for full-event simulation at the LHC and their application to pileup description. In *Journal of Physics: Conference Series*, volume 1525. IOP Publishing, 2020.

K. T. Matchev and P. Shyamsundar. Uncertainties associated with GAN-generated datasets in high energy physics. *arXiv preprint arXiv:2002.06307*, 2020.

C. J. Matheus and L. A. Rendell. Constructive induction on decision trees. In *IJCAI*, volume 89, pages 645–650, 1989.

R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11 (3-4):365–396, 2010a.

R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11 (3-4):365–396, 2010b.

D. A. Melis and T. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pages 7775–7784, 2018.

C. Mencar. Interpretability of fuzzy systems. In *International Workshop on Fuzzy Logic and Applications*, pages 22–35. Springer, 2013.

M. Mestayer, K. Adhikari, R. Bennett, S. Bueltmann, T. Chetry, S. Christo, M. Cook IV, R. Cuevas, G. Dodge, L. El Faasi, et al. The CLAS12 drift chamber system. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 959:163518, 2020.

R. S. Michalski. On the quasi-minimal solution of the general covering problem. *Proceedings of the 5th International Symposium on Information Processing*, 3:125–128, 1969.

G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

J. F. Miller and S. L. Harding. Cartesian genetic programming. In *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, pages 2701–2726, 2008.

P. Miquilini, R. C. Barros, V. V. de Melo, and M. P. Basgalupp. Enhancing discrimination power with genetic feature construction: A grammatical evolution approach. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3824–3831. IEEE, 2016.

S. Mohseni, N. Zarei, and E. D. Ragan. A survey of evaluation methods and measures for interpretable machine learning. *arXiv preprint arXiv:1811.11839*, 2018.

G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

D. J. Montana. Strongly typed genetic programming. *Evolutionary computation*, 3 (2):199–230, 1995.

E. Mugambi, A. Hunter, G. Oatley, and L. Kennedy. Polynomial-fuzzy decision tree structures for classifying medical data. *Knowledge-Based Systems*, 2(17):81–87, 2004.

M. Muharram and G. D. Smith. Evolutionary constructive induction. *IEEE transactions on knowledge and data engineering*, 17(11):1518–1528, 2005.

H. Murase, H. Nagashima, S. Yonezaki, R. Matsukura, and T. Kitakado. Application of a generalized additive model (GAM) to reveal relationships between environmental factors and distributions of pelagic fish and krill: a case study in Sendai Bay, Japan. *ICES Journal of Marine Science*, 66(6):1417–1424, 2009.

F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. S. Turaga. Learning feature engineering for classification. In *IJCAI*, pages 2529–2535, 2017.

K. Neshatian, M. Zhang, and M. Johnston. Feature construction and dimension reduction using genetic programming. In *Australasian Joint Conference on Artificial Intelligence*, pages 160–170. Springer, 2007.

K. Neshatian, M. Zhang, and P. Andreae. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation*, 16(5):645–661, 2012.

W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip. Efficient clustering of uncertain data. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 436–445. IEEE, 2006.

A.-p. Nguyen and M. R. Martínez. MonoNet: Towards interpretable models by learning monotonic features. *arXiv preprint arXiv:1909.13611*, 2019.

H. Nori, S. Jenkins, P. Koch, and R. Caruana. InterpretML: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy sets and systems*, 138(2):221–254, 2003.

J. L. Olmo, J. R. Romero, and S. Ventura. A grammar based ant programming algorithm for mining classification rules. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

F. E. Otero and A. A. Freitas. Improving the interpretability of classification rules discovered by an ant colony algorithm. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 73–80, 2013.

F. E. Otero and A. A. Freitas. Improving the interpretability of classification rules discovered by an ant colony algorithm: extended results. *Evolutionary computation*, 24(3):385–409, 2016.

F. E. Otero, M. M. Silva, A. A. Freitas, and J. C. Nievola. Genetic programming for attribute construction in data mining. In *European Conference on Genetic Programming*, pages 384–393. Springer, 2003.

G. Pagallo. Learning DNF by decision trees. In *IJCAI*, volume 89, pages 639–644, 1989.

S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

N. Papernot and P. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. An ant colony algorithm for classification rule discovery. In *Data mining: A heuristic approach*, pages 191–208. IGI Global, 2002.

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

W. Pedrycz and Z. A. Sosnowski. The design of decision trees in the framework of granular data and their application to software quality models. *Fuzzy Sets and Systems*, 123(3):271–290, 2001.

G. Peyré and M. Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

R. Pierrard, J.-P. Poli, and C. Hudelot. A fuzzy Close algorithm for mining fuzzy association rules. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 88–99. Springer, 2018.

A. Pierrot and Y. Goude. Short-term electricity load forecasting with generalized additive models. *Proceedings of ISAP power*, 2011.

S. Piramuthu and R. T. Sikora. Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications*, 36(2):3401–3406, 2009.

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

B. Qin, Y. Xia, and F. Li. DTU: a decision tree for uncertain data. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 4–15. Springer, 2009a.

B. Qin, Y. Xia, S. Prabhakar, and Y. Tu. A rule-based classification algorithm for uncertain data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1633–1640. IEEE, 2009b.

Z. Qin and J. Lawry. Decision tree learning with fuzzy labels. *Information Sciences*, 172(1-2):91–129, 2005.

J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J. R. Quinlan. Generating production rules from decision trees. In *International Joint Conference on Artificial Intelligence*, page 304–307. Morgan Kaufmann Publishers Inc., 1987.

J. R. Quinlan. Learning logical definitions from relations. *Machine learning*, 5(3): 239–266, 1990.

J. R. Quinlan. MDL and categorical theories (continued). In *Machine Learning Proceedings 1995*, pages 464–470. Elsevier, 1995.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Elsevier, 2014.

J. P. Ralston and B. Pire. Femtophotography of protons to nuclei with deeply virtual compton scattering. *Physical Review D*, 66(11):111501, 2002.

E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi. GSA: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248, 2009.

A. Ratle and M. Sebag. Avoiding the bloat with stochastic grammar-based genetic programming. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 255–266. Springer, 2001a.

A. Ratle and M. Sebag. Grammar-guided genetic programming and dimensional consistency: application to non-parametric identification in mechanics. *Applied Soft Computing*, 1(1):105–118, 2001b.

I. Redko, N. Courty, R. Flamary, and D. Tuia. Optimal transport for multi-source domain adaptation under target shift. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 849–858, 2019.

M. S. Reis and P. M. Saraiva. Integration of data uncertainty in linear regression and process optimization. *AIChE journal*, 51(11):3007–3019, 2005.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

T. J. Ross. *Fuzzy logic with engineering applications*. John Wiley & Sons, 2005.

Y. Rubner, C. Tomasi, and L. J. Guibas. The Earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

P. J. Sadowski, D. Whiteson, and P. Baldi. Searching for Higgs boson decay modes with deep learning. *Advances in neural information processing systems*, 27:2393–2401, 2014.

L. Sánchez and J. Otero. Boosting fuzzy rules in classification problems under single-winner inference. *International journal of intelligent systems*, 22(9):1021–1034, 2007.

M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

E. Schubert and P. J. Rousseeuw. Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms. In *International Conference on Similarity Search and Applications*, pages 171–187. Springer, 2019.

A. G. Sébert and J.-P. Poli. Fuzzy rule learning for material classification from imprecise data. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 62–73. Springer, 2018a.

A. G. Sébert and J.-P. Poli. Material classification from imprecise chemical composition: Probabilistic vs possibilistic approach. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2018b.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

L. S. Shafti and E. Pérez. Fitness function comparison for GA-based feature construction. In *Conference of the Spanish Association for Artificial Intelligence*, pages 249–258. Springer, 2007.

L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2 (28):307–317, 1953.

D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

M. Smith and L. Bull. Improving the human readability of features constructed by genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1694–1701, 2007.

M. G. Smith and L. Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3): 265–281, 2005.

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

T. Sousa, A. Silva, and A. Neves. Particle swarm based data mining algorithms for classification tasks. *Parallel computing*, 30(5-6):767–783, 2004.

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

M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440, 2008.

J.-N. Sulzmann and J. Fürnkranz. A comparison of techniques for selecting and combining class association rules. In *Proceedings of the LeGo'08: From Local Patterns to Global Models, ECML/PKDD 2008 Workshop*, pages 87–93, 2008.

B. Sun and K. Saenko. Deep CORAL: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. *arXiv preprint arXiv:1511.05547*, 2015.

I. M. A. O. Swesi. Feature clustering for PSO-based feature construction on high-dimensional data. *Journal of Information and Communication Technology*, 18(4): 439–472, 2020.

I. M. A. O. Swesi and A. A. Bakar. Recent developments on evolutionary computation techniques to feature construction. In *Asian Conference on Intelligent Information and Database Systems*, pages 109–122. Springer, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

M. Tami, M. Clausel, E. Devijver, A. Dulac, E. Gaussier, S. Janaqi, and M. Chebre. Uncertain trees: Dealing with uncertain inputs in regression trees. *arXiv preprint arXiv:1810.11698*, 2018.

A. L. Tarca, V. J. Carey, X.-w. Chen, R. Romero, and S. Drăghici. Machine learning and its applications to biology. *PLoS Comput Biol*, 3(6):e116, 2007.

Y. Teng and A. Choromanska. Invertible autoencoder for domain adaptation. *Computation*, 7(2):20, 2019.

E. Tjoa and C. Guan. A survey on explainable artificial intelligence (XAI): Towards medical XAI. *arXiv preprint arXiv:1907.07374*, 2019.

B. Tran, B. Xue, and M. Zhang. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing*, 8(1):3–15, 2016a.

B. Tran, M. Zhang, and B. Xue. Multiple feature construction in classification on high-dimensional data using GP. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016b.

B. Tran, B. Xue, and M. Zhang. Genetic programming for multiple-feature construction on high-dimensional classification. *Pattern Recognition*, 93:404–417, 2019.

S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee. Decision trees for uncertain data. *IEEE transactions on knowledge and data engineering*, 23(1):64–78, 2009.

R. Turrisi, R. Flamary, A. Rakotomamonjy, and M. Pontil. Multi-source domain adaptation via weighted joint distributions optimal transport. *arXiv preprint arXiv:2006.12938*, 2020.

G. Tutz and H. Binder. Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics*, 62(4):961–971, 2006.

A. T. Tzallas, I. Tsoulos, M. G. Tsipouras, N. Giannakeas, I. Androulidakis, and E. Zaitseva. Classification of EEG signals using feature creation produced by grammatical evolution. In *2016 24th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2016.

E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

M. Ungaro, G. Angelini, M. Battaglieri, V. Burkert, D. Carman, P. Chatagnon, M. Contalbrigo, M. Defurne, R. De Vita, B. Duran, et al. The CLAS12 GEANT4 simulation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 959:163422, 2020.

H. Vafaie and K. De Jong. Genetic algorithms as a tool for restructuring feature space representations. In *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 8–11. IEEE, 1995.

D. E. Van de Vlag and A. Stein. Incorporating uncertainty via hierarchical classification using fuzzy decision trees. *IEEE Transactions on geoscience and remote sensing*, 45(1):237–245, 2006.

M. Vanderhaeghen, P. A. Guichon, and M. Guidal. Deeply virtual electroproduction of photons and mesons on the nucleon: Leading order amplitudes and power corrections. *Physical Review D*, 60(9):094017, 1999.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.

A. Vidon. *Probing the proton structure through deep virtual Compton scattering at COMPASS, CERN*. PhD thesis, Université Paris-Saclay, 2019.

C.-H. Wang, W.-H. Lee, and C.-T. Pang. Applying fuzzy FP-Growth to mine fuzzy association rules. *World Academy of Science, Engineering and Technology*, 65:956–962, 2010.

X. Wang, B. Chen, G. Qian, and F. Ye. On the optimization of fuzzy decision trees. *Fuzzy Sets and Systems*, 112(1):117–125, 2000.

X. Wang, Y. Jin, M. Long, J. Wang, and M. I. Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1951–1961, 2019.

R. Weber. Fuzzy-ID3: A class of methods for automatic knowledge acquisition. *International Conference on Fuzzy Logic and Neural Networks*, pages 265–268, 01 1992.

P. A. Whigham, G. Dick, J. Maclaurin, and C. A. Owen. Examining the "best of both worlds" of grammatical evolution. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1111–1118, 2015.

Wikipedia. Standard model, 2020. URL https://en.wikipedia.org/wiki/Standard_Model. [Online; accessed 28-July-2020].

G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

S. Wood and M. S. Wood. Package 'mgcv'.

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

S. N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2 edition, 2017.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

B. Xue, M. Zhang, Y. Dai, and W. N. Browne. PSO for feature construction and binary classification. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 137–144, 2013.

D.-S. Yang, L. A. Rendell, and G. Blix. A scheme for feature construction and a comparison of empirical methods. In *IJCAI*, pages 699–704. Citeseer, 1991.

H.-J. Yang, B. P. Roe, and J. Zhu. Studies of boosted decision trees for MiniBooNE particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 555(1-2): 370–385, 2005.

S. Yazdani, J. Shanbehzadeh, and E. Hadavandi. MBCGP-FE: A modified balanced Cartesian genetic programming feature extractor. *Knowledge-Based Systems*, 135: 89–98, 2017.

C. Yildiz and E. Alpaydin. Omnivariate decision trees. *IEEE Transactions on Neural Networks*, 12(6):1539–1546, 2001.

Y. Yuan and M. J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and systems*, 69(2):125–139, 1995.

L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1(1):3–28, 1978.

M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

J. Zhang, F. Fogelman-Soulié, and C. Largeron. Towards automatic complex feature engineering. In *International Conference on Web Information Systems Engineering*, pages 312–322. Springer, 2018.

Y. Zhang and P. I. Rockett. A generic optimising feature extraction method using multiobjective genetic programming. *Applied Soft Computing*, 11(1):1087–1097, 2011.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

M. Zięba, S. K. Tomczak, and J. M. Tomczak. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert systems with applications*, 58:93–101, 2016.

V. Ziegler, N. Baltzell, F. Bossù, D. Carman, P. Chatagnon, M. Contalbrigo, R. De Vita, M. Defurne, G. Gavalian, G. Gilfoyle, et al. The CLAS12 software framework and event reconstruction. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 959:163472, 2020.

H.-J. Zimmermann. *Fuzzy set theory—and its applications*. Springer Science & Business Media, 2011.

G. Zweig. *An SU(3) model for strong interaction symmetry and its breaking. Version 2*, volume 1, pages 22–101. 1964.

P. Zyla et al. Review of Particle Physics. *PTEP*, 2020(8):083C01, 2020.

Appendix A

Introduction to fuzzy logic

A.1 Principle and operators

Fuzzy logic has been introduced by Zadeh [1965] and extends Boolean logic with partial truth information. In Boolean logic, an element x belongs or not to a subset A of an ensemble E: $x \in A$ or $x \notin A$. However, in fuzzy logic, each element x gets a *membership degree* to the subset A given by the *membership function* $f_A : E \to [0, 1]$. The principle is illustrated on Figure A.1 with $E = [0, 10]$: the left plot represents a crisp set $A = \{x \geq 5\}$, while the right plot represents a fuzzy set with a linear membership function in the interval $[4, 6]$. The transition does not have to be linear, although it is faster to compute. Shapes of membership functions can be triangular (as on Figure A.1), trapezoidal, Gaussian among others. References for fuzzy logic are the books of Ross [2005] and Zimmermann [2011].

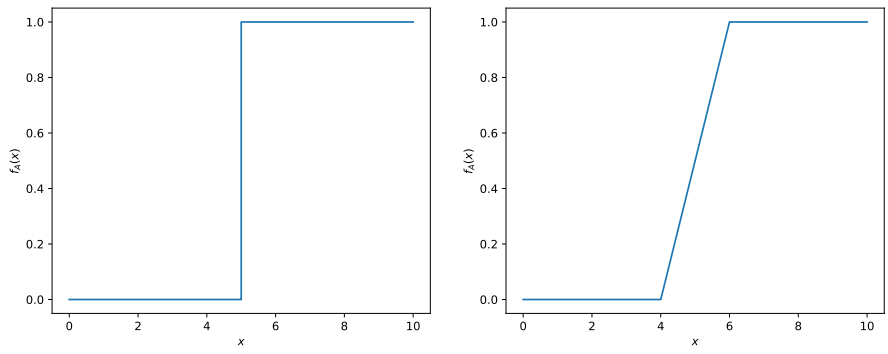


FIGURE A.1: Illustration of fuzzy logic with $E = [0, 10]$.
On the left: a crisp set $A = \{x \geq 5\}$.
On the right, a fuzzy set $f_A(x) = 0$ if $x < 4$, $f_A(x) = 1$ if $x > 6$,
$f_A(x) = \frac{x}{2} - 2$ if $x \in [4, 6]$.

The classical AND and OR operators are extended as well to fuzzy logic: while probabilistic operators are the multiplication for the AND and the addition for the OR, Zadeh uses the minimum as AND operator and the maximum as OR operator. In this way, the membership functions of intersection, union, and complement of fuzzy sets can be derived:

- $f_{A \cap B} : x \mapsto \min\left(f_A(x), f_B(x)\right)$;
- $f_{A \cup B} : x \mapsto \max\left(f_A(x), f_B(x)\right)$;
- $f_{\bar{A}} : x \mapsto 1 - f_A(x)$.

Note that some common properties of Boolean logic are no longer maintained: the intersection of a subset and its complement may be not empty ($A \cap \bar{A} \neq \varnothing$), and their union can have a membership function non equal to 1 ($\exists\, x \in E,\ f_{A \cup \bar{A}}(x) \neq 1$).

A.2 Reasoning with fuzzy logic

"IF ... THEN" rules can be defined with fuzzy sets, for example:

$$\text{IF} \quad x \in A_1 \quad \text{OR} \quad y \in B_1 \quad \text{THEN} \quad z \in C_1$$

with A_1, B_1 and C_1 being fuzzy subsets of A, B and C the respective domains of x, y and z. The Mamdani inference process is illustrated on Figure A.2, although other generalizations of the logic implication exist [Ross, 2005, Chapter 5]. With given x and y, the membership degree of z in C is:

$$\mu_C(z) = \min\left[\max(f_{A_1}(x), f_{B_1}(y)), f_{C_1}(z)\right]. \tag{A.1}$$

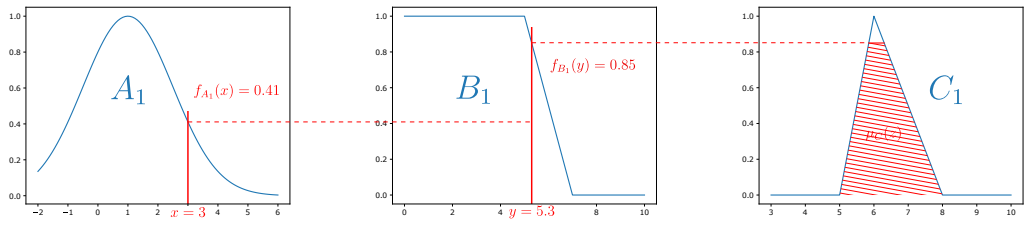


FIGURE A.2: Mamdani inference of a single rule: IF $x \in A_1$ OR $y \in B_1$ THEN $z \in C_1$ for $x = 3$ and $y = 5.3$.

A complete rule base usually covers the entire definition domain of the input variables. A fuzzy partitioning is defined such as the one on Figure A.3 for y in B. An example of a rule set is:

$$\begin{array}{lllll} \text{IF} & x \in A_1 & \text{OR} & y \in B_1 & \text{THEN} \quad z \in C_1 \\ \text{IF} & x \in A_2 & \text{OR} & y \in B_3 & \text{THEN} \quad z \in C_2 \\ & & \text{IF} & y \in B_2 & \text{THEN} \quad z \in C_3. \end{array}$$

The inference is then made by computing the disjunction between the rules, i.e. taking the maximum (union) of all rule outputs, as illustrated on Figure A.4. The final z is *defuzzified* either by taking for instance the mean of maximum (MoM) or the center of gravity (COG).

Another popular type of fuzzy rule is Takagi-Sugeno fuzzy rule (as opposed to Mamdani fuzzy rules illustrated above). Instead of having a fuzzy set as output, Takagi-Sugeno fuzzy rules output a function of the inputs, for instance:

$$\text{IF} \quad x \in A_1 \quad \text{OR} \quad y \in B_1 \quad \text{THEN} \quad z = f(x, y)$$

where f is any real function.

A.3 Fuzzy expert systems

From the inference principles described above, a fuzzy expert system can be designed: as illustrated by Figure A.5, a fuzzy system first fuzzifies the inputs (i.e. computes the

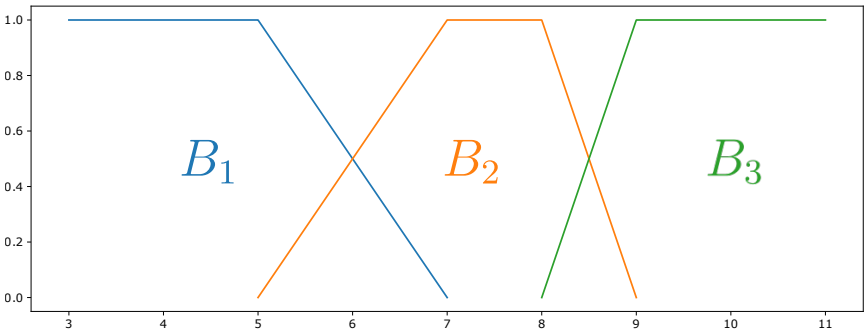


FIGURE A.3: A fuzzy trapezoidal partition of B.

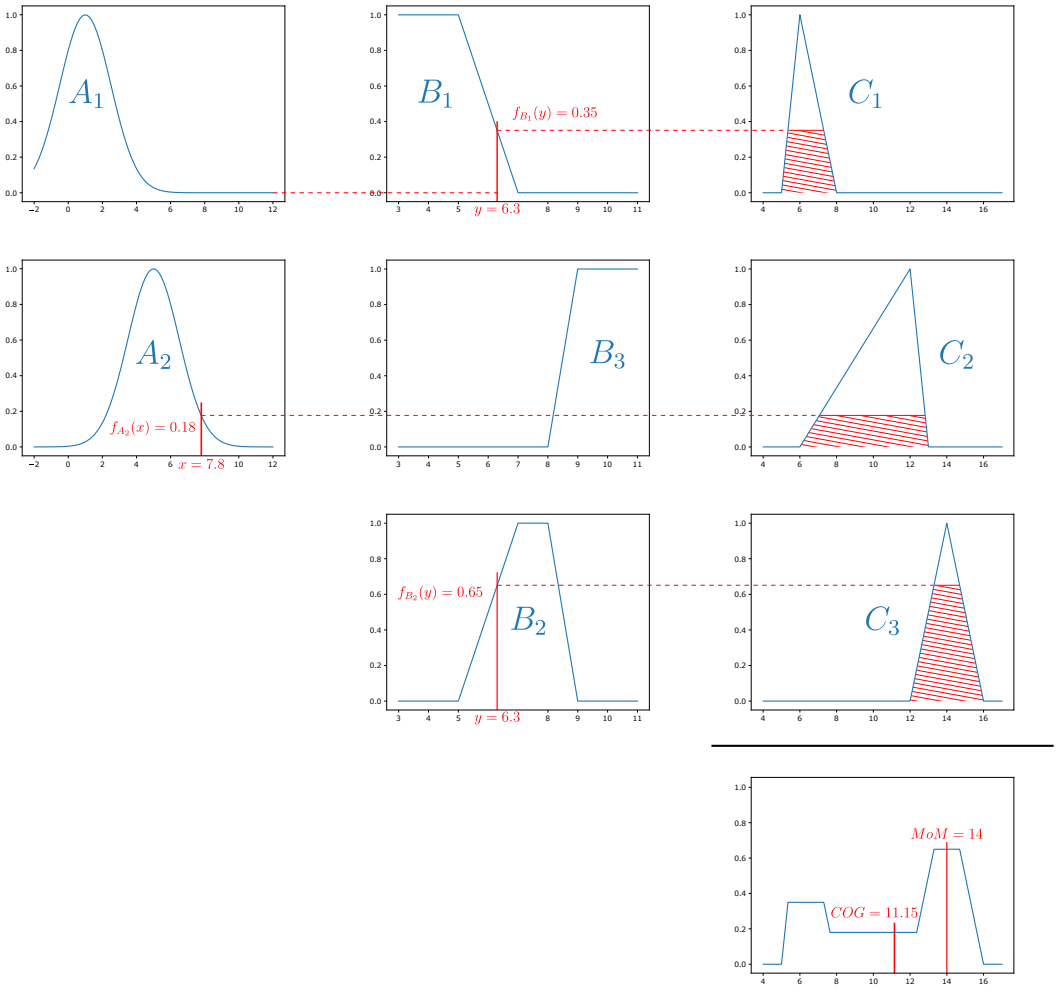


FIGURE A.4: Induction of a fuzzy rule set with $x = 7.8$ and $y = 6.3$. One line corresponds to one rule in the set.

membership degrees to every subset of the fuzzy partition), computes the inference, and defuzzifies the outputs (i.e. returns a crisp output value). All of these steps pick the necessary elements in the knowledge base, containing the fuzzy partitionings and rules that can be either determined by domain experts or automatically derived.

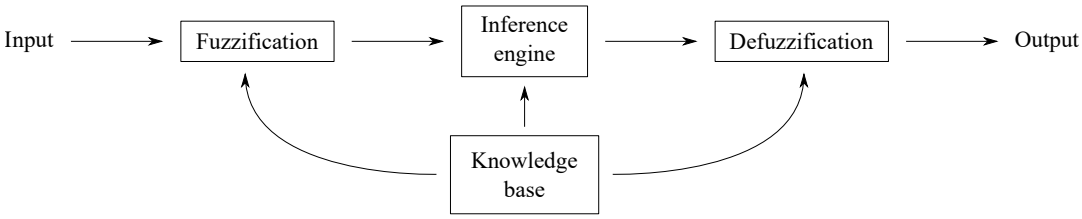


FIGURE A.5: Fuzzy expert systems.

A.4 Handling imprecisions

Instead of crisp values, such systems are also able to handle data with imprecisions, i.e. fuzzy numbers. A fuzzy number x is associated to a membership function μ_x defined on the definition domain A. The membership degree of a fuzzy number x to a fuzzy subset A_1 is given by:

$$\widetilde{\mu}_{A_1}(x) = \max_t \left[\min(f_{A_1}(t), \mu_x(t))\right]. \tag{A.2}$$

An alternative is to take the integral of the product instead of the maximum over the t variable of the minimum. The two approaches are compared by Sébert and Poli [2018b] for the induction of fuzzy decision trees. Once the inputs have been fuzzified (i.e. once their membership degrees to every subset in the partition have been computed), the inference is performed the same way as for crisp inputs.

A.5 Advantages of fuzzy logic in machine learning

Hüllermeier [2015] discusses the assets of fuzzy logic for machine learning, and acknowledges its interest to express uncertainty. According to him, fuzzy logic is not the best method to enhance algorithms such as support vector machines (SVM) (a Gaussian kernel is more interesting) or nearest neighbors (a weighting scheme is the same), with the notable exception of decision trees and rules. Arrieta et al. [2020] underline the ability of fuzzy logic to better handle uncertainty and to improve understandability in decision rules.

Appendix B

Exploiting data imprecisions

Real-world data often comprise uncertainty in measurement, as they come from sensors or detectors with limited resolutions, and as the experimental conditions are not always exactly known. Uncertainty and imprecision handling is difficult with statistical models such as neural networks that perform simple operations on input variables. However, these uncertainties may be in themselves a valuable information. For instance in a classification problem, if the measures are very accurate, the classifier should be very certain of its decision. On the opposite, if a variable is subject to big errors of measurement, the classifier can rely on other variables to guide the final output and should not be biased by a misleading crisp mean value.

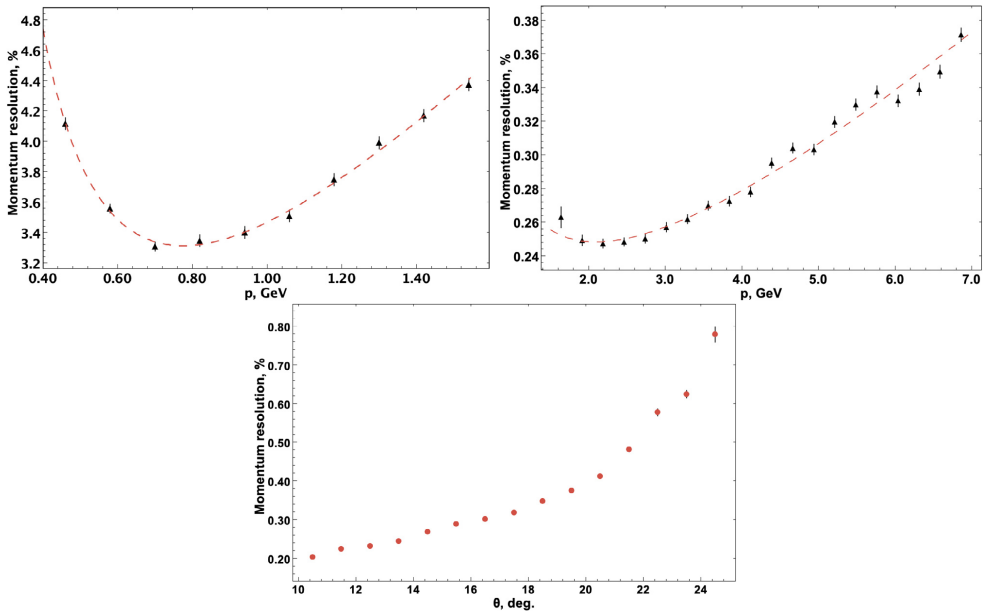


FIGURE B.1: Resolutions of the tracking on simulation: as function of the momentum p for protons in the central tracker (top left) and for pions in the forward tracker (top right), as function of the scattering angle θ in the forward tracker (bottom) [Ziegler et al., 2020].

In HEP notably, particle detectors are subject to noise and introduce uncertainties in event reconstruction since particles interact with the detectors they cross. These imperfections are related to the detector resolutions that are not always well known, but also to physics phenomena and geometry of the experiment. Therefore, for each momentum that is measured for a particle, there is an associated error of measurement that notably depends on the geometric location of the particle inside the detector.

Figure B.1 illustrates this phenomenon for CLAS12: particles with low momentum p interact more with matter thus loosing energy faster; therefore, the resolution is degraded at low p due to these multiple scatterings. On the opposite, particles with large momenta are less deviated by the magnetic field, therefore the measurement of their curvature radius is more imprecise. This leads to a higher uncertainty on their momentum. Regarding the scattering angle θ, the intensity of the magnetic field is higher at low θ angles, making the curvature measurement easier and the momentum computation more precise.

Our objective is to use this additional information in an interpretable classifier, able to distinguish between relevant signal events and background events.

Notably, fuzzy logic treats uncertainties in measurement as imprecisions in the data (see Appendix A). We are thus interested in fuzzy models and notably FURIA and fuzzy C4.5 trees. A number of algorithms exist that are fuzzy in their functioning, but that do not necessarily take imprecise data as inputs. In this study, we adapt FURIA and fuzzy C4.5 to handle imprecise data, and apply these adapted algorithms to CLAS12 event selection.

Section B.1 recalls the representation of imprecision in fuzzy logic and lists a few articles that exploit data imprecision. Sections B.2 and B.3 present the adaptations respectively of crisp and fuzzy C4.5 and FURIA to handle imprecisions in data. CLAS12 imprecisions are computed in B.4 and experiments conducted in B.5. Finally, a discussion is proposed in section B.6.

B.1 Background on the use of imprecisions in transparent models

Instead of a vector of crisp numerical values, imprecise data consists of a vector of fuzzy numbers. These fuzzy numbers are represented by their membership function μ_x over the definition domain of x, as illustrated on Figure B.2.

Two alternatives coexist to handle imprecisions in fuzzy logic: the possibilistic and the probabilistic approaches.

- In the possibilistic approach, the usual Zadeh operators of fuzzy logic are used. Namely, the minimum between two fuzzy numbers computes a conjunction and the maximum a disjunction. An imprecise variable is represented by a normalized fuzzy number so that its maximum reaches 1.
- In the probabilistic approach, the product of two fuzzy numbers computes a conjunction, and a disjunction is the sum of these fuzzy numbers minus the conjunction ($A + B - A \times B$). Imprecise variables are represented by fuzzy numbers whose integrals equal 1.

When the data are imprecise and when the imprecision is known and characterized, the classical machine learning algorithms hardly manage this type of attribute. Machine learning usually tackles uncertainty in the target variable or in the model itself [Campagner et al., 2020].

Nevertheless, Czarnecki and Podolak [2013] augment the number of instances by sampling many data points from imprecise measures.

$$x_1 = 4.5, \qquad x_2 = -1.7, \qquad x_3 = -0.38, \qquad x_4 = 42$$

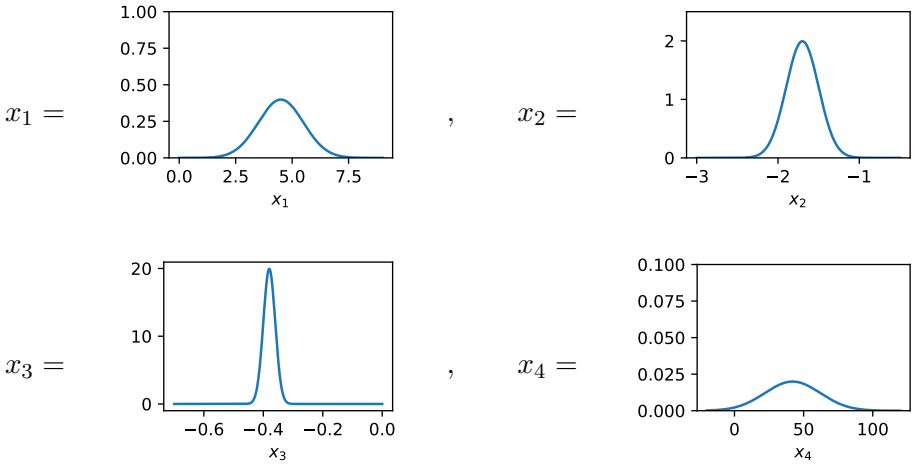


FIGURE B.2: On the top: an instance without imprecisions. All variables are crisp numbers. On the bottom: an instance with imprecisions. The variables are represented as fuzzy numbers in the probabilistic framework.

A few works directly model data uncertainty in the models. Reis and Saraiva [2005] integrate data uncertainty in multivariate regression models. Huang et al. [2012] propose a support vector regression model with uncertain data. Several works propose uncertain clustering Cormode and McGregor [2008], Kriegel and Pfeifle [2005], Ngai et al. [2006] with a distance between probability distributions instead of usual Euclidean distances.

Some works propose to represent uncertain data under the form of probability distributions, and adapt the induction of decision trees Qin et al. [2009a], Tami et al. [2018] or rule bases Qin et al. [2009b] to handle them. The criteria used in such models are adapted: a probabilistic entropy for decision trees, a probabilistic information gain for rules.

Using fuzzy logic, Maher and Clair [1993] do not change the induction of the classical ID3 decision tree but adapt its reasoning: the imprecisions about some inputs are propagated through the tree using fuzzy logic and the possibility theory to produce an imprecise output. A few works exploit the possibility theory to adapt the induction of fuzzy ID3 to uncertain data [Maher and Clair, 1993, Borgelt et al., 1996, Jenhani et al., 2008]. A Bayesian hierarchical model is used by Van de Vlag and Stein [2006] to perform clustering before inducing a fuzzy ID3 tree to classify geographical objects.

Sébert and Poli [2018a] propose a fuzzy clustering of the input variables using k-medoids [Schubert and Rousseeuw, 2019] with a dissimilarity measure between two probabilistic distributions. Then, the fuzzy ID3 is induced using a fuzzified version of the entropy criterion. This new criterion is specifically adapted for probabilistic distributions. The resulting tree is a hybrid approach mixing probabilistic distributions and fuzzy logic while the natural approach would have been possibilistic decision trees [Maher and Clair, 1993, Borgelt et al., 1996, Jenhani et al., 2008]. However, the hybrid approach seems to overcome the possibilistic one [Sébert and Poli, 2018b], at

least on their experiments on nuclear physics data.

B.2 Adaptation of crisp and fuzzy C4.5 algorithms to imprecise data

In addition to the fuzzy ID3 algorithm, we propose to compare against the crisp and fuzzy C4.5 algorithms that we adapt to be able to handle imprecise data. Imprecise data are represented by the Gaussian probability distributions for each variable. We focus on the probabilistic management of imprecision as it provides better results for the use case of Sébert and Poli [2018b].

B.2.1 Crisp C4.5

Instead of following a single branch, instances propagate through the branches with different weights according to their membership degree to each branch. As illustrated on Figure B.3, the membership degrees of an instance x to the children nodes of a parent node N are given by:

$$f_{\leq\alpha}(x) = f_N(x) \int_{-\infty}^{\alpha} \mu_x(t)\,\mathrm{d}t, \tag{B.1}$$

$$f_{>\alpha}(x) = f_N(x) \int_{\alpha}^{+\infty} \mu_x(t)\,\mathrm{d}t \tag{B.2}$$

with $f_{\leq\alpha}(x)$ and $f_{>\alpha}(x)$ the membership degrees of example x respectively to the left and right child node, α the split threshold of the parent node N, $f_N(x)$ the membership degree of x to the parent node N, and μ_x the membership function of x.

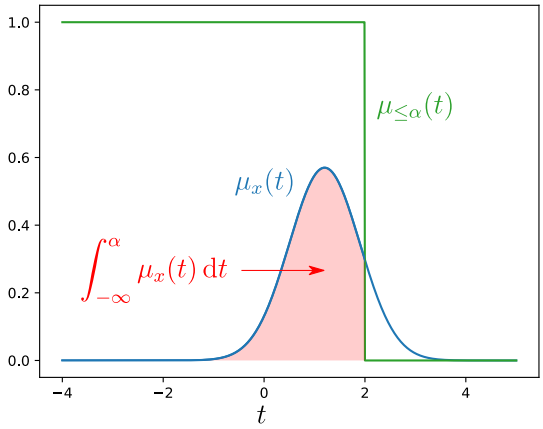


FIGURE B.3: Computation of the membership degree of the fuzzy example x to an antecedent of type $\leq$ with threshold α.

In the case of crisp C4.5 decision tree, we adapt the entropy measure of the classical algorithm to consider the imprecisions of the data. Using precise data, the information gain IG of an attribute A is the difference between the entropy of the current node and the weighted sum of the entropies of the children nodes:

$$IG = H(X) - \sum_i p_i H(X_i). \tag{B.3}$$

$$H(X) = -\sum_k p_k(X) \log_2 p_k(X). \tag{B.4}$$

X is the data contained in the current node, and the X_i are the data subsets that would be created splitting along the tested attribute A. p_i is the proportion of instances in child node i. k covers the space of data labels, and $p_k(X)$ is the proportion of class k amongst the dataset X.

Considering data imprecisions, the proportions p_k are fuzzified to use the membership degrees $f_N(x_i)$ of the examples $x_i \in X$ to the current node N:

$$p_k(X) = \frac{\sum_{i,x_i \in k} f_N(x_i)}{\sum_i f_N(x_i)}. \tag{B.5}$$

B.2.2 Fuzzy C4.5

While the computation of the information gain remains the same as for crisp C4.5, taking membership degrees into account, the computation of these membership degrees gets more complicated: indeed, the split of a node is itself fuzzified. Therefore, the membership degree of an example to a node becomes the integral of the product between the membership function of the example and the membership function of the node itself, as illustrated on Figure B.4:

$$f_{\leq\alpha}(x) = f_N(x) \int_{-\infty}^{+\infty} \mu_x(t)\mu_{\leq\alpha}(t)\,\mathrm{d}t, \tag{B.6}$$

$$f_{>\alpha}(x) = f_N(x) \int_{-\infty}^{+\infty} \mu_x(t)\mu_{>\alpha}(t)\,\mathrm{d}t \tag{B.7}$$

with same notations as above and $\mu_{\leq\alpha}(t)$ and $\mu_{>\alpha}(t)$ the membership functions of the left and right children nodes. Notably we have:

$$\forall t \in \mathbb{R}, \quad \mu_{\leq\alpha}(t) + \mu_{>\alpha}(t) = 1. \tag{B.8}$$

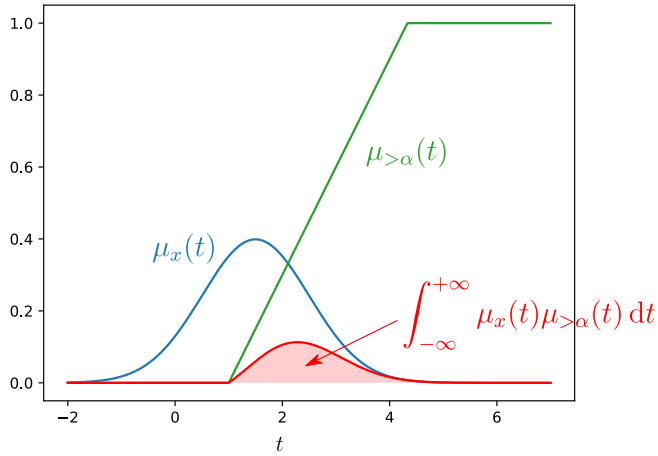


FIGURE B.4: Computation of the membership degree of the fuzzy example x to a fuzzy antecedent.

B.3 Adaptation of FURIA to imprecise data

As in the previous section, the different metrics and heuristics used in FURIA have to be adapted to our representation of imprecise data by Gaussian probability distribution.

FURIA follows the rule covering principle, namely the algorithm progressively adds rules until all instances of a given class are *covered* by the rules. A particular rule is grown until all instances of other classes are no longer *covered* by the rule. After rule bases have been built for every class, optimization is performed: a replacement rule and a revision rule are proposed for each already built rule, and the best among the three is retained for the final rule base. Pruning is performed at this step, using the subset of examples of the pruning dataset that are not *covered* by the subsequent rules of the rule base. Overall, the notion of *coverage* is critical in FURIA but is intrinsically a Boolean, crisp notion. To adapt it to imprecise data, we define that an instance is covered by an antecedent, rule or rule base if its coverage degree to the antecedent, rule or rule base is non zero. The coverage degree of an imprecise instance x to an antecedent A writes:

$$f_A(x) = \int_{-\infty}^{\alpha} \mu_x(t)\,\mathrm{d}t \text{ if the antecedent is } \leq \alpha, \tag{B.9}$$

$$f_A(x) = \int_{\alpha}^{+\infty} \mu_x(t)\,\mathrm{d}t \text{ if the antecedent is } \geq \alpha \tag{B.10}$$

with α the split threshold of the antecedent and μ_x the membership function of x (the Gaussian probability distribution). Then, the coverage degree of an imprecise instance x to a rule writes:

$$f_R(x) = \prod_i f_{A_i}(x) \tag{B.11}$$

as the product of the coverage degrees by all antecedents of the rule. Finally, the coverage degree to a rule base writes as the sum of the coverage degrees over the rules of the base:

$$f_{RB}(x) = \sum_j f_{R_j}(x). \tag{B.12}$$

Using the proposed definition of coverage degree, the different steps requiring a crisp notion of coverage are adapted:

- FURIA adds rules for a given class until all instances of the class are at least partially covered (namely $\forall$ positive x, $f_{RB}(x) > 0$);
- a particular rule is grown until all instances of other classes are entirely not covered (namely $\forall$ negative x, $f_{RB}(x) = 0$);
- pruning is performed on all instances that are not entirely covered by the subsequent rules in the base; their coverage degrees to the subsequent rules are taken into account to weight the pruning criteria.

FURIA uses many criteria at the various steps of the induction (see 2.2.2): the FOIL information gain to grow the rules, the pruning metric, the rule set error and the minimum description length during optimization, the purity for fuzzification. For

instance, the FOIL information gain writes:

$$IG = p_r \left(\log_2 \left(\frac{p_r}{p_r + n_r} \right) - \log_2 \left(\frac{p}{p+n} \right) \right) \tag{B.13}$$

with p_r and n_r the number of positive and negative examples covered by the rule, and p and n the number of positive and negative examples covered by the default rule (i.e. current rule, without adding a new antecedent).

With precise data, p, p_r, n and n_r are integers, while they are real numbers for imprecise data. The membership function is therefore introduced in the computation of these variables:

$$p = \sum_{x \in Pos} f_R(x) \qquad p_r = \sum_{x \in Pos} f_R(x) f_A(x) \tag{B.14}$$

$$n = \sum_{x \in Neg} f_R(x) \qquad n_r = \sum_{x \in Neg} f_R(x) f_A(x) \tag{B.15}$$

R is the rule already learnt during the current growing phase and A the candidate antecedent evaluated by the information gain calculation.

The membership functions $f_R(x)$ and $f_A(x)$ are computed following the same principle as for decision trees: Equation (B.1) before the rules are fuzzified, and Equation (B.6) afterwards.

These measures of the instances coverage are used in a similar manner for the other criteria in FURIA: the pruning metric and the fuzzification purity, and for the computation of rule stretching measures and of the certainty factors for each rule.

B.4 Computation of CLAS12 imprecisions

We know that the reconstruction of the energies and three-momenta of particles does not give a perfect image of the particles. The geometry of the experiment and the detector resolutions are encoded into the GEMC software that serves to imitate the real detectors in simulations [Ungaro et al., 2020].

The performance of an ideal spectrometer and of the reconstruction can be characterized using the simulation since we have the knowledge of the generated particles. The imprecisions mainly depend on the momentum p and of the θ angle between the beam axis and the particle's direction, as seen on Figure B.1. For charged particles such as the electron and the proton, tracking will determine the resolution. However, photons are only detected in calorimeters.

To estimate the data imprecisions and try to exploit them in classifiers, we derive the theoretical imprecisions from simulated data. We compute the errors on the momentum p, θ and ϕ angles as function of the momentum p and θ and ϕ angles. Figure B.5 illustrates the errors between reconstructed and generated particles as function of the momentum p. The resolutions on photons energies are largely worse than for other particles, and often exceed 5%. The expected resolution for the PCAL and EC calorimeters is indeed $\frac{10\%}{\sqrt{E\ (\text{GeV})}}$ [Burkert et al., 2020]. On the opposite, the resolution of the drift chambers in the forward detector is expected to be comprised between 0.5 and 1.5%.

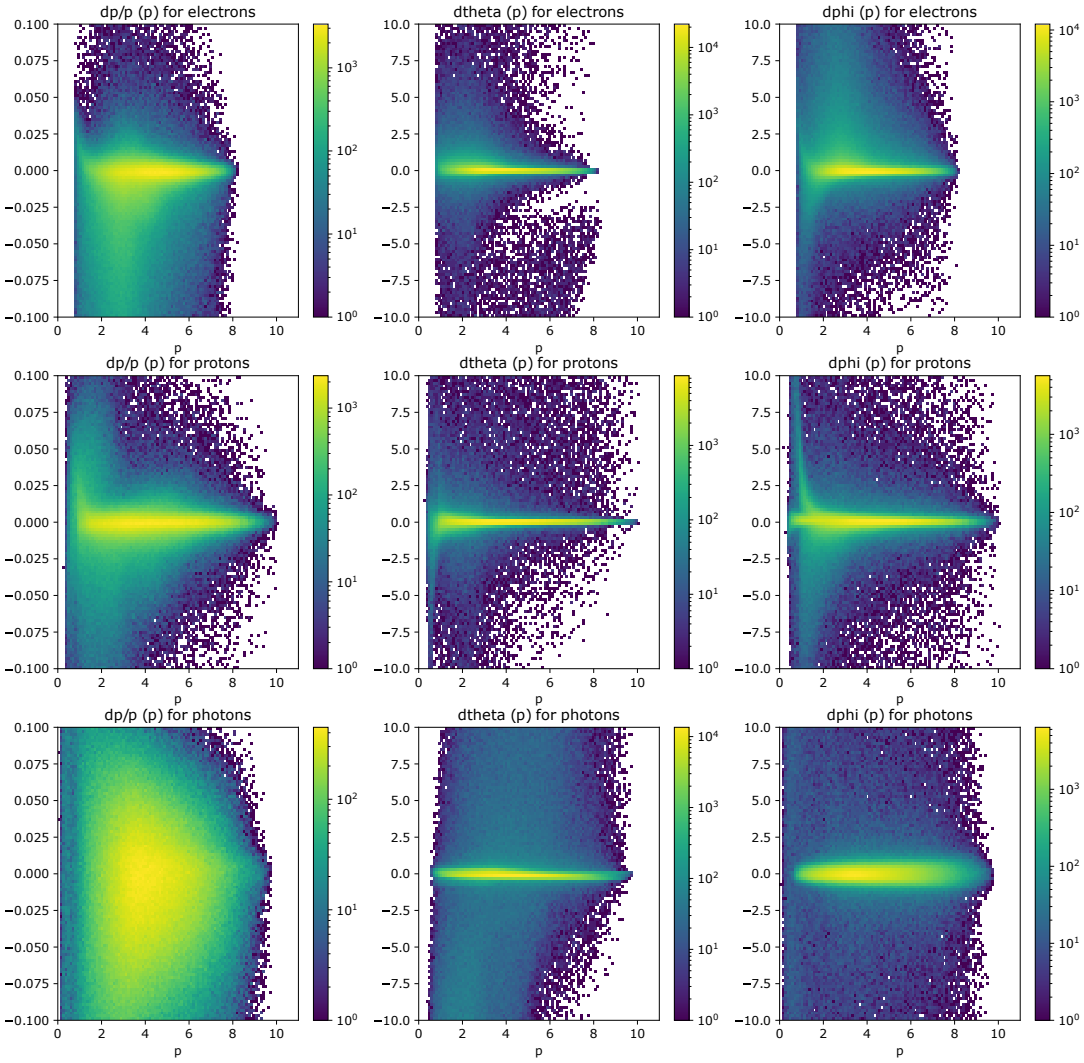


FIGURE B.5: Distributions of errors on (p, θ, ϕ) for electrons, protons and photons as function of the momentum p.

Since the errors are symmetrical in ϕ, we compute imprecisions that depend only on p and θ. The steps are as follows:

1. we divide the simulated data in 10 bins for p and 10 bins for θ so that the data is equally divided into the resulting bins;
2. in each bin, we compute the relative errors on p and the absolute errors on θ and ϕ;
3. we take the standard deviation of $\frac{dp}{p}$, $d\theta$ and $d\phi$ in each bin, which is the final imprecision for a given bin;
4. covariances between p, θ and ϕ are also computed to be able to compute the imprecisions on the Cartesian coordinates dp_x, dp_y and dp_z.

Here biases are not considered, i.e. when the mean of the errors is not 0. This must be corrected on the data themselves and not through fuzzy solutions.

However, these computations require knowing beforehand the detector resolutions. Here, they are implemented in the simulation software as estimations of the real

detectors resolutions. Moreover, the calibration may be sub-optimal in reality and induce reconstruction errors, which is not the case in pure simulation. For now, we conduct experiments using the simulations. Therefore, this method is not directly applicable to real data unless the Monte Carlo simulation shows perfect agreement with data.

B.5 Experiments

Table B.1 presents the accuracies averaged over five folds along with their standard deviations obtained when training different models while taking account or not of data imprecisions. In addition, Table B.1 also displays the scores while considering only the imprecisions on the electron and proton, and not on the photons. This last score is not presented for FURIA, for which the implementation does not currently authorize having mixture of precise and imprecise input features.

TABLE B.1: Accuracy scores obtained with different usages of data imprecisions, for crisp C4.5, fuzzy C4.5 (std or fibo versions), and FURIA. The first line displays the scores without imprecisions, the second line with imprecisions on the electron and proton only, and the last line with all imprecisions (including those on the photons).

	Crisp C4.5	Fuzzy C4.5 std	Fuzzy C4.5 fibo	FURIA
No imp.	0.651 ± 0.004	0.659 ± 0.009	0.658 ± 0.008	0.604 ± 0.016
e and p	0.642 ± 0.008	0.647 ± 0.009	0.645 ± 0.010	(not implemented)
All imp.	0.612 ± 0.011	0.612 ± 0.010	0.612 ± 0.009	0.570 ± 0.010

In all cases, using the knowledge about imprecisions degrades significantly the classification score. Moreover, the imprecisions on the photons are the one contributing the most to the score drop: in average, 74% of the decrease is due to photons in all versions of C4.5.

To further investigate, Figure B.6 displays the ROC curves for the crisp and fuzzy C4.5 trees. However, FURIA is not adapted to the usage of ROC curves.

The drop in performance, observed through accuracies, is visible again here. However, starting from $\simeq 10\%$ of false positives, the difference is not significant any more. Since it is the region of interest for DVCS event selection, this is a direction for further research.

Finally, we manually vary the imprecision, from 0% to 100% of its value. We obtain a progressive decrease in accuracy as displayed on Figure B.7. The ROC curves are displayed on Figure B.8. The same progressive drop is observed.

However, zooming on the ROC curves gives Figure B.9: on this zoom, the curves seem to mix. Progressively, the curves taking into account an increasing degree of imprecisions surpass the no imprecisions curve. It seems that the imprecisions add valuable information at small false positives rates, but that the model benefits from a moderate imprecision information.

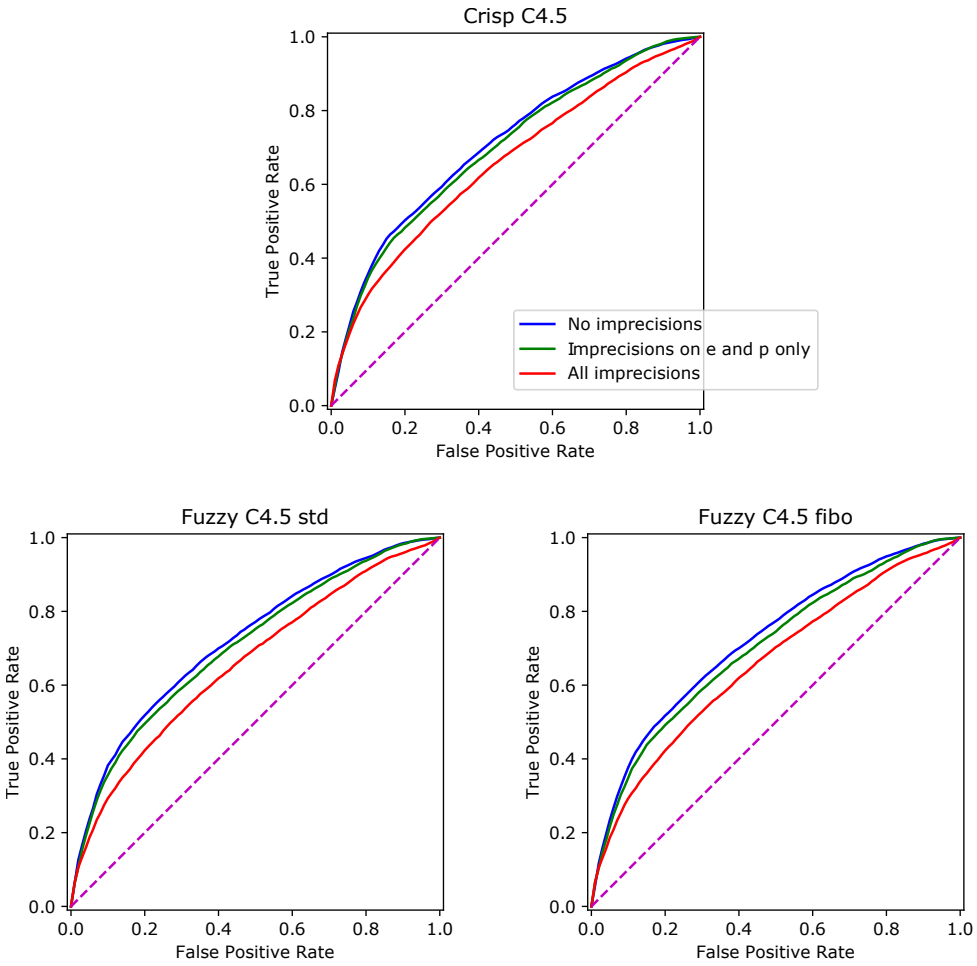


FIGURE B.6: ROC curves of crisp C4.5 (left), fuzzy C4.5 std (middle) and fuzzy C4.5 fibo (right) while taking into account or not data imprecisions.

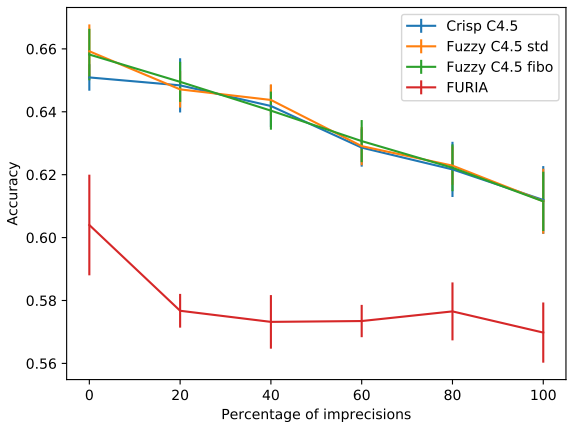


FIGURE B.7: Accuracy as function of the imprecision degree, for the crisp and fuzzy C4.5 versions and for FURIA.

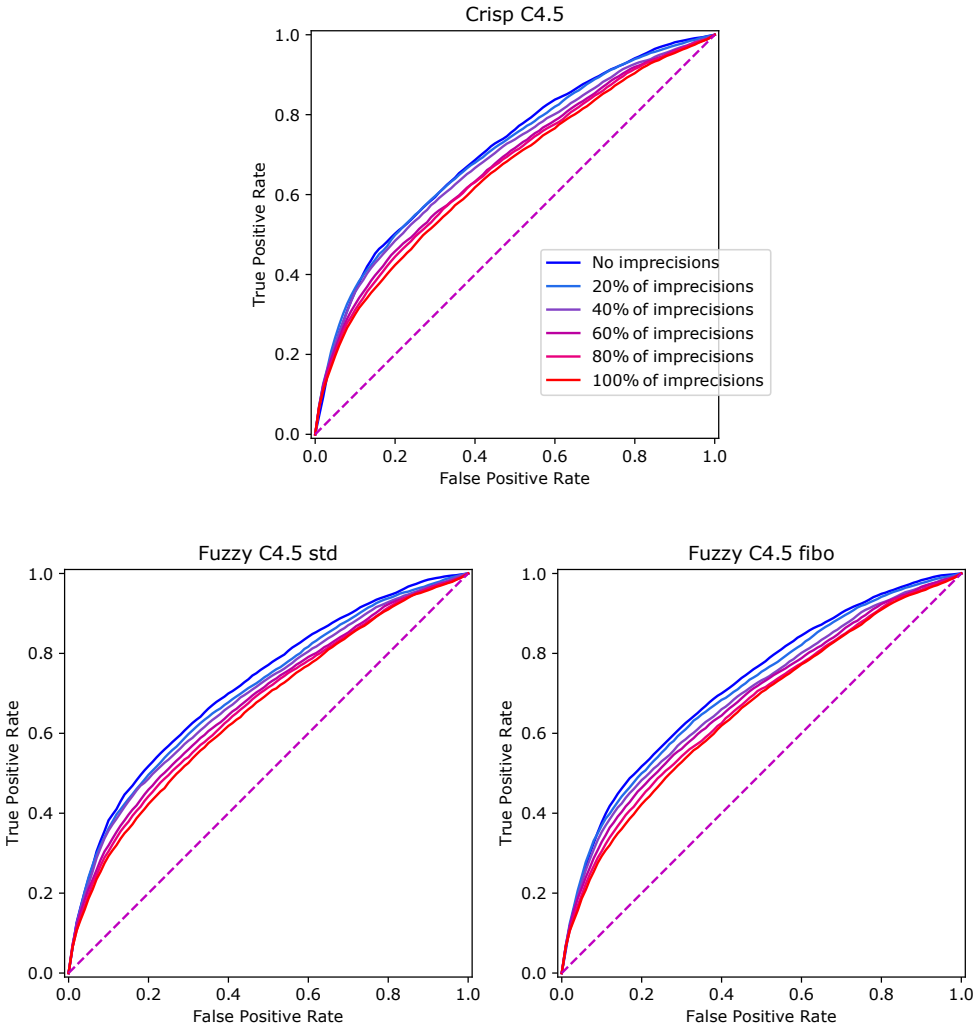


FIGURE B.8: ROC curves crisp C4.5 (left), fuzzy C4.5 std (middle) and fuzzy C4.5 fibo (right) while taking into account data imprecisions at different degrees, from blue (no imprecisions) to red (100% of imprecisions).

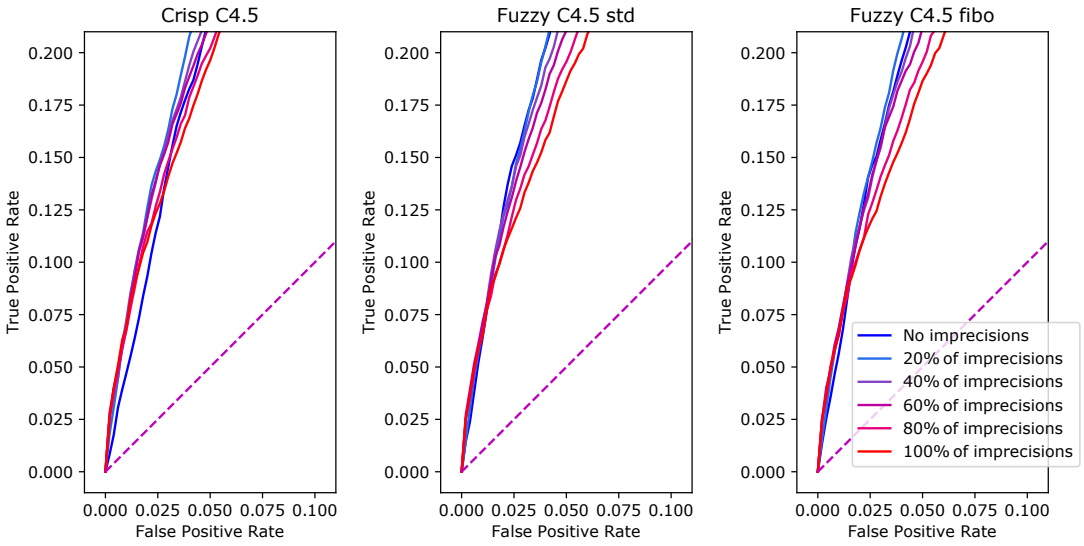


FIGURE B.9: Zoom on the ROC curves of Figure B.8.

B.6 Discussion

There are two opposite visions about the usage of imprecisions:

- Adding prior information is supposed to help the model improving its classification. In this case, we give the knowledge about which variables are the most imprecise and therefore the least reliable to use. Split thresholds can also be adapted to handle more or less imprecise values on both sides. In addition, knowing data imprecision permits the model to temper its decision when the inputs are too imprecise. Imprecisions should help decrease false positives and false negatives at extreme regions of the ROC curve by softening radical classifications.
- However, imprecisions are useful to prioritize the most precise variables to perform the classification, when there are several variables carrying the same information: otherwise, there is no other choice than to use an imprecise variable if no other correlated variable is available. In addition, the three coordinates of the particle's three-momenta are probably correlated, and the imprecisions similar among them. Therefore, maybe the knowledge of imprecisions can not improve the classification.

We observe that the usage of imprecisions in transparent machine learning models does not necessarily help the classification. However, there is a hope for improvement in the small false positive rate region, where using imprecisions seems to be able to increase the true positive rate. The framework to embed imprecisions into the machine learning models may not be adapted: the model should be able to choose to which extent it is useful to exploit imprecisions, while the proposed method imposes the procedure to use imprecise information. This forced functioning can be the cause for the degraded classification performances. Otherwise, the imprecisions themselves could be badly modeled: a Gaussian with a smaller standard deviation is perhaps more subtle, or other probability distributions might be more appropriate.

One main point about the usage of imprecisions is the correlation between variables. A perspective would be to search for other variables as combinations of the base ones, so that their imprecisions are cancelled thanks to the correlations of the base variables but so that their discriminative power stays high.

However, one main limitation to this work is the determination of the imprecisions themselves: as is, it requires knowing the generated particles' three-momenta, before going through the detectors and passing the reconstruction process. Therefore, the procedure demands a better knowledge of the detector resolutions and a validation of the simulation. This can be solved notably using *domain adaptation*, which is tackled in chapter 7.

Appendix C

Experimental datasets

Part II describes several machine learning algorithms that have been developed in this thesis. To assess their interest for other physics problems than CLAS12 data classification, other physics-related public datasets are used. A description of the statistics and associated physics problem is provided for each dataset that is used for experiments in part II.

C.1 CLAS12

Although the generation method of this dataset is detailed in chapter 3, a summary of the statistics and features is provided here.

Summary of the physics problem

At the CLAS12 experiment at Jefferson Laboratory, an electron beam scatters off protons at rest in the lab frame. The objective is to discriminate between the DVCS events whose final state is composed of an electron, a proton, and a photon noted γ, and the π^0 production events that have a similar final state, except that the photon is replaced by a π^0. The later immediately decays into two correlated photons. One of them may not be detected, mimicking a DVCS event. On the other side, accidental photons may appear during a DVCS event, which will then have two or more photons in its final state while remaining a signal event.

Available features

Up to five particles of the output state are kept to form the feature set: one electron, one proton, and up to three photons ranked by missing mass $ep \rightarrow ep\gamma$. In total, the 35 available features are the three-dimensional momentum (namely mass times speed of the particle) for each identified particle, expressed in two coordinate systems, plus the three-vector as is: $p_x, p_y, p_z, p_T, \theta, \phi, \mathbf{p}$.

Grammar

The grammar for the CLAS12 dataset is displayed on Figure C.1. It involves energies in GeV, angles in radians and also handles operations on vectors (on the three-momenta of the output particles).

The transition matrix is presented in Table C.1.

```
<start> ::= <E> | <E2> | <A> | <F>
<E>   ::= <E> + <E> | <E> − <E> | <E> × <F> | <E> ÷ <F>
   | sqrt(<E2>) | norm(<M>) | <component>(<M>) | <termE>
<E2> ::= <E2> + <E2> | <E2> − <E2> | <E2> × <F> | <E2> ÷ <F>
   | <E> × <E> | square(<E>) | dot(<M>,<M>) | <termE2>
<A>   ::= <A> + <A> | <A> − <A> | <A> × <F> | <A> ÷ <F>
   | angle(<M>,<M>) | <termA>
<F>   ::= <F> + <F> | <F> − <F> | <F> × <F> | <F> ÷ <F>
   | <E> ÷ <E> | <E2> ÷ <E2> | <A> ÷ <A> | sqrt(<F>) | square(<F>)
   | cos(<A>) | sin(<A>) | tan(<A>) | <termF>
<M>   ::= <M> + <M> | <M> − <M> | <termM>
<component> ::= get_x | get_y | get_z
```

FIGURE C.1: Grammar used for the CLAS12 dataset. E stands for a 1D momentum or energy in GeV, E2 for a squared momentum or energy in GeV^2, A for an angle in radians, F for a unitless real number, M for a three-momentum of unit GeV. <termX> means a terminal of type X, namely a base feature or a constant.

TABLE C.1: Transition matrix for the CLAS12 dataset. The probabilities are displayed for the next possible operations (as columns) given the previous operation (as row). Operations that are not listed as rows have a uniform transition probability distribution. Operations that are not listed as columns for a given previous operation cannot be selected as next operation (probability 0). The notations are the same than in the grammar (Figure C.1).

Return type: {E: 0.5, A: 0.2, F: 0.1}.

	E + E	E - E	E × F	E ÷ F	sqrt(E2)
E + E	0.1	0.1	0.1	0.1	0.6
E - E	0.225	0.225	0.25	0.2	0.1

	E + E	E - E	E × F	E ÷ F	norm(M)
square(E)	0.5	0.1	0.07	0.03	0.3

	E2 + E2	E2 - E2	square(E)	E × E
E2 + E2	0.4	0.15	0.4	0.05
E2 - E2	0.2	0.07	0.7	0.03
sqrt(E2)	0.7	0.25	0	0.05

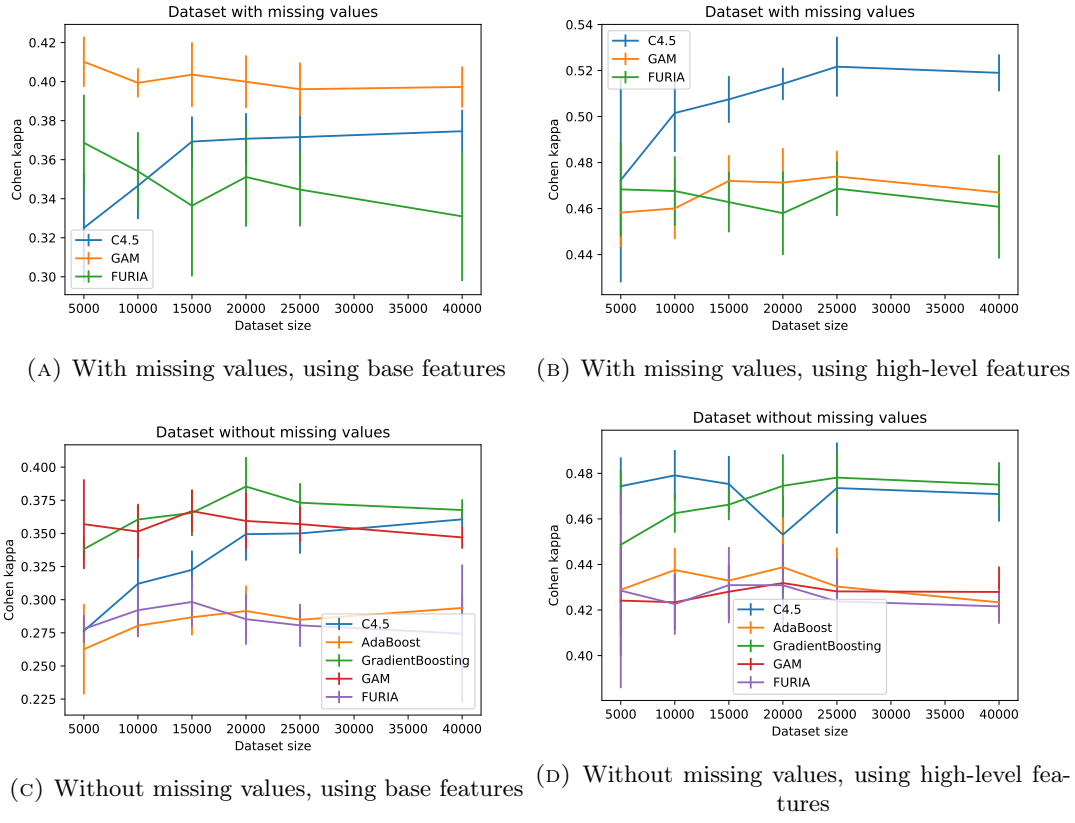
	E + E	E - E	E × F	E ÷ F	sqrt(E2)
E × F	0.15	0.15	0.35	0.3	0.05
E ÷ F	0.15	0.15	0.35	0.3	0.05

	F + F	F - F	F × F	square(F)	cos(A)	sin(A)	tan(A)
E × F	0.025	0.025	0.025	0.025	0.375	0.375	0.15
E ÷ F	0.025	0.025	0.025	0.025	0.1	0.1	0.7

	M + M	M - M
norm(M)	0.9	0.1

Size of the dataset Two variants of this dataset are used depending on the considered learning algorithm. Indeed, this dataset comprises missing values: many events (i.e. instances) involve less than three photons, which creates missing values for these photons-related features. Some of the algorithms presented in the following do not handle missing values. Consequently, two different datasets are considered: one of 25000 instances comprising missing values for some of them, and the other of 14730 instances with no missing value, i.e. with three photons associated to each event. The major drawback of this second dataset is that the phase space is biased: the machine learning algorithm will not be able to learn to discriminate between single photon events. For algorithms that cannot handle missing values, several separate algorithms need to be trained for each subconfiguration.

These sizes have been chosen regarding baseline scores obtained by a few transparent machine learning models, as displayed on Figure C.2. Since part II is using feature construction to enhance the performances of these models, baselines using common high-level variables used by physicists are also computed on subfigures C.2b and C.2d.



(A) With missing values, using base features

(B) With missing values, using high-level features

(C) Without missing values, using base features

(D) Without missing values, using high-level features

FIGURE C.2: Evolution of the Cohen's kappa metric on a 5-fold cross-validation for different dataset sizes.

C.2 Higgs

This simulated data [Adam-Bourdarios et al., 2014] is publicly available on the Open Data platform of CERN[1], and was the subject of a Kaggle challenge[2] in 2014.

[1] opendata.cern.ch/record/328
[2] kaggle.com/c/higgs-boson

TABLE C.2: Overview of the different processes present in the Higgs dataset. ${}^{h}\tau$ denotes a hadronic τ and ${}^{l}\tau$ a leptonic τ.

Signal	$H \to {}^{h}\tau + {}^{l}\tau \to h\ \nu + lep\ \nu\ \nu$
Background	$W \to {}^{h}\tau + {}^{l}\tau \to h\ \nu + lep\ \nu\ \nu$
	$t\ t \to lep + {}^{h}\tau \to lep + h\ \nu$
	$W \to lep + {}^{h}\tau \to lep + h\ \nu$

Physics problem

At CERN, two protons collide head-on with each other and Higgs particles are notably produced out of the collisions. The objective of the dataset is to detect Higgs bosons decaying into two τ-particles. One of the τs (leptonic τ) subsequently decays into one lepton (electron or muon) and two neutrinos, and the other τ (hadronic τ) creates hadronic jets and one neutrino. Three other processes form the background:

- The first background process involves the decay of a Z boson instead of a Higgs boson. Identically to the Higgs boson, the Z boson can decay into two τ leptons: one leptonic τ decaying into one lepton and two neutrinos and one hadronic τ decaying into hadrons and one neutrino.
- The second background process comes from two top quarks (noted t). They decay into a lepton and a hadronic τ (producing hadrons and one neutrino).
- The third background process consists in the decay of a W boson into a lepton and a hadronic τ, similarly to the second background process.

All of these processes are summarized in Table C.2. The main difficulty comes from the fact that neutrinos cannot be measured in the detectors. Only the "missing momentum" can be computed, namely the direction and energy of the sum of the missed neutrinos, based on energy and momentum conservation in the detector. Only events comprising only one lepton (electron or muon) and one hadronic τ are retained for the dataset.

Available features

The public dataset comprises 17 "primitive" features and 13 "derived" features. The latter include manually designed high-level features that involve multiple particles. Because the objective of this part is to perform automatic feature construction, these derived features are removed in the used dataset, keeping only the 17 primitive features. These primitive features include: the transverse momentum p_T, the polar angle θ and the azimuthal angle ϕ for the reconstructed hadronic τ, the lepton, the leading jet (i.e. the jet with the largest transverse momentum), the subleading jet (i.e. with the second largest momentum). In addition, the energy and ϕ angle of the missing momentum are included, as well as the total transverse energy in the detector, the number of jets, and the scalar sum of the transverse momentum of the jets.

Grammar

The grammar for the Higgs dataset, displayed on Figure C.3, is quite similar to the one of the CLAS12 dataset, since the same kind of input variables are present: energies in GeV and angles in radians. Since we are using the dataset as provided by CERN, we do not add redundant variables such as the Cartesian coordinates or the three-vector.

```
<start> ::= <E> | <E2> | <A> | <F>
<E>   ::= <E> + <E> | <E> − <E> | <E> × <F> | <E> ÷ <F>
   | sqrt(<E2>) | <termE>
<E2> ::= <E2> + <E2> | <E2> − <E2> | <E2> × <F> | <E2> ÷ <F>
   | <E> × <E> | square(<E>) | <termE2>
<A>   ::= <A> + <A> | <A> − <A> | <A> × <F> | <A> ÷ <F> | <termA>
<F>   ::= <F> + <F> | <F> − <F> | <F> × <F> | <F> ÷ <F>
   | <E> ÷ <E> | <E2> ÷ <E2> | <A> ÷ <A> | sqrt(<F>) | square(<F>)
   | cos(<A>) | sin(<A>) | tan(<A>) | <termF>
```

FIGURE C.3: Grammar used for the Higgs dataset. E stands for a 1D momentum or energy in GeV, E2 for a squared momentum or energy in GeV^2, A for an angle in radians, F for a unitless real number. <termX> means a terminal of type X, namely a base feature or a constant.

The transition matrix (Table C.3) is also similar to the one of the CLAS12 dataset.

TABLE C.3: Transition matrix for the Higgs dataset. The probabilities are displayed for the next possible operations (as columns) given the previous operation (as row). Operations that are not listed as rows have a uniform transition probability distribution. Operations that are not listed as columns for a given previous operation cannot be selected as next operation (probability 0). The notations are the same than in the grammar (Figure C.3).

Return type: {E: 0.5, A: 0.2, F: 0.1}.

	E + E	E - E	E × F	E ÷ F	sqrt(E2)
E + E	0.1	0.1	0.1	0.1	0.6
E - E	0.225	0.225	0.25	0.2	0.1

	E + E	E - E	E × F	E ÷ F
square(E)	0.8	0.1	0.07	0.03

	E2 + E2	E2 - E2	square(E)	E × E
E2 + E2	0.4	0.15	0.4	0.05
E2 - E2	0.2	0.07	0.7	0.03
sqrt(E2)	0.7	0.25	0	0.05

	E + E	E - E	E × F	E ÷ F	sqrt(E2)
E × F	0.15	0.15	0.35	0.3	0.05
E ÷ F	0.15	0.15	0.35	0.3	0.05

	F + F	F - F	F × F	square(F)	cos(A)	sin(A)	tan(A)
E × F	0.025	0.025	0.025	0.025	0.375	0.375	0.15
E ÷ F	0.025	0.025	0.025	0.025	0.1	0.1	0.7

Size of the dataset

The dataset consists of more than 800000 events including about 280000 signal events. Only 100000 events are kept for the experiments with the same ratio between the two classes (34% of signal events).

C.3 $\tau \to 3\mu$

One of the research projects of the Large Hadron Collider beauty (LHCb) experiment at CERN is to try to observe the decay of the τ^- particle into three muons [LHCb collaboration, 2015]. The problem also led to a Kaggle challenge in 2015[3].

Physics problem

The studied physics process is the decay of a τ^- into three muons: $\tau^- \to \mu^+\mu^-\mu^-$. This decay is not supposed to happen according to the Standard Model: the lepton flavor is not conserved. An observation of such an event would mean a violation of the latter and consequently a sign of new physics. The particularity of this dataset is that background events are taken from real data measured at LHCb, so a wide variety of processes are present. Signal events are simulated to complete the dataset.

Available features

38 features from the original dataset are kept for the feature construction study. These features include geometrical information about the muons, such as their transverse momentum p_T and polar angle θ, but also their impact parameter, isolation variables, etc. Unfortunately, the designers of this dataset do not provide the azimuthal angles ϕ, which prevents the construction of several high-level variables that are yet relevant.

Grammar

The grammar for the $\tau \to 3\mu$ dataset (displayed on Figure C.4) is quite different from the ones of CLAS12 and Higgs since it involves distances in addition to energies and angles.

```
<start> ::= <E> | <E2> | <A> | <F>
<E>   ::= <E> + <E> | <E> − <E> | <E> × <F> | <E> ÷ <F>
   | sqrt(<E2>) | <termE>
<E2>  ::= <E2> + <E2> | <E2> − <E2> | <E2> × <F> | <E2> ÷ <F>
   | <E> × <E> | square(<E>) | <termE2>
<A>   ::= <A> + <A> | <A> − <A> | <A> × <F> | <A> ÷ <F> | <termA>
<F>   ::= <F> + <F> | <F> − <F> | <F> × <F> | <F> ÷ <F>
   | <E> ÷ <E> | <E2> ÷ <E2> | <A> ÷ <A> | sqrt(<F>) | square(<F>)
   | cos(<A>) | sin(<A>) | tan(<A>) | <termF>
```

FIGURE C.4: Grammar used for the $\tau \to 3\mu$ dataset. E stands for a 1D momentum or energy in GeV, E2 for a squared momentum or energy in GeV^2, A for an angle in radians, D for a distance in centimeters, F for a unitless real number. <termX> means a terminal of type X, namely a base feature or a constant.

Size of the dataset

The dataset comprises more than 67500 events including about 40000 signal events. It should be noted that the dataset originally comes with an agreement dataset and a correlation dataset, to ensure a few constraints are respected for the further physics analysis: for instance, the learning algorithm should not learn the difference between

[3] kaggle.com/c/flavours-of-physics

simulated and real events but rather the difference between signal and background events. The two additional datasets are not used in the following studies.

C.4 MAGIC

This dataset comes from the UCI (University of California at Irvine) machine learning repository [Dua and Graff, 2017].

Physics problem

The Major Atmospheric Gamma-ray Imaging Cherenkov (MAGIC) Telescope indirectly observes gamma rays from their interaction with the Earth's atmosphere. Electronic showers are produced by the entry of gamma particles in the atmosphere, and are seen from the ground by the MAGIC telescope. The objective is to recognize from the projection of the shower onto the telescope surface whether this shower originates from a primary gamma source (signal event) or from cosmic rays (background).

Available features and size of the dataset

The 10 numerical features of the dataset describe the geometry of the reconstructed ellipse that has been seen in a simulated telescope. 19020 events are available, including 65% of signal gamma events.

No grammar is used for this dataset (therefore, the feature construction performed in part II is unconstrained) since we are not expert of the physics behind this experiment.

C.5 Summary

Table C.4 summarizes the characteristics of the presented datasets.

Table C.4: Characteristics of the datasets used for the experiments. The balance degree is the normalized entropy of the label vector.

Name	Size	Features	Balance degree
Higgs	100000	17	0.93
CLAS12	25000 / 14730	35	1.00 / 0.96
$\tau\mu^3$	67553	38	0.96
MAGIC	19020	10	0.94

Appendix D

Model hyperparameters

The following hyperparameters are used in all experiments.

C4.5

- the minimum number of instances per leaf is 2;
- the minimal information gain to divide a node is 10^{-4};
- no post-pruning is performed, because our implementation did not match the one of Weka.

Fuzzy C4.5 std In addition to the C4.5 parameters, the α parameter to multiply the standard deviation of the attribute to determine the width of the fuzzy transition is set to $\alpha = 0.125659549$ (10% of the Gaussian distribution).

Fuzzy C4.5 Fibo In addition to the C4.5 parameters, the Fibonacci search of the width β of the fuzzy transition is performed with 5 iterations.

CART The minimum number of instances per leaf is 2.

AdaBoost The individual classifiers are the CART detailed above with maximal depth 3. 50 classifiers constitute the tree ensemble. There is not shrinkage of the classifiers (namely the learning rate is 1). The boosting uses class probabilities.

GradientBoosting 50 classifiers constitute the tree ensemble, the individuals being CART regressors of maximal depth 3. The criterion for the CART regressors is the mean squared error. The global loss is the logistic regression loss, with learning rate 0.1.

FURIA The minimum number of instances per split is 2. Two optimizations steps are performed using three folds.

GAM The baselines are established with the backfitting algorithm and one shape function per input feature. For FCGAM, the boosting algorithm described in 6.2 is used. For splines, the used technique to fit the shape functions is REML. The functions are B-splines with 20 knots. For neural network, a fully-connected network of two layers of size 100 with rectified linear unit as activation is used to fit the shape functions. The optimization is made with an Adam optimizer with $\beta_1 = 0.9$ and

$\beta_2 = 0.999$ for maximum 200 epochs. The default regularization is 0.0001 but is tuned in 6.2.

Appendix E

Additional experiments on embedded feature construction in tree-based models

E.1 Fuzzy C4.5: std version

TABLE E.1: Cohen's kappa score with different feature construction methods embedded into a fuzzy C4.5: std version. The number of built features leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.342 ± 0.022	0.394 ± 0.004
MC	0.422 ± 0.023	0.413 ± 0.028
Downgraded GP	0.482 ± 0.019 (100)	0.467 ± 0.016 (100)
GP	**0.498 ± 0.016 (100)**	**0.478 ± 0.015 (100)**
	$\tau \rightarrow 3\mu$	MAGIC
Baseline (without feature construction)	0.542 ± 0.004	0.675 ± 0.011
MC	0.536 ± 0.018	0.655 ± 0.024
Downgraded GP	**0.553 ± 0.017 (100)**	**0.678 ± 0.017 (100)**
GP	0.537 ± 0.032 (15)	0.660 ± 0.027 (100)

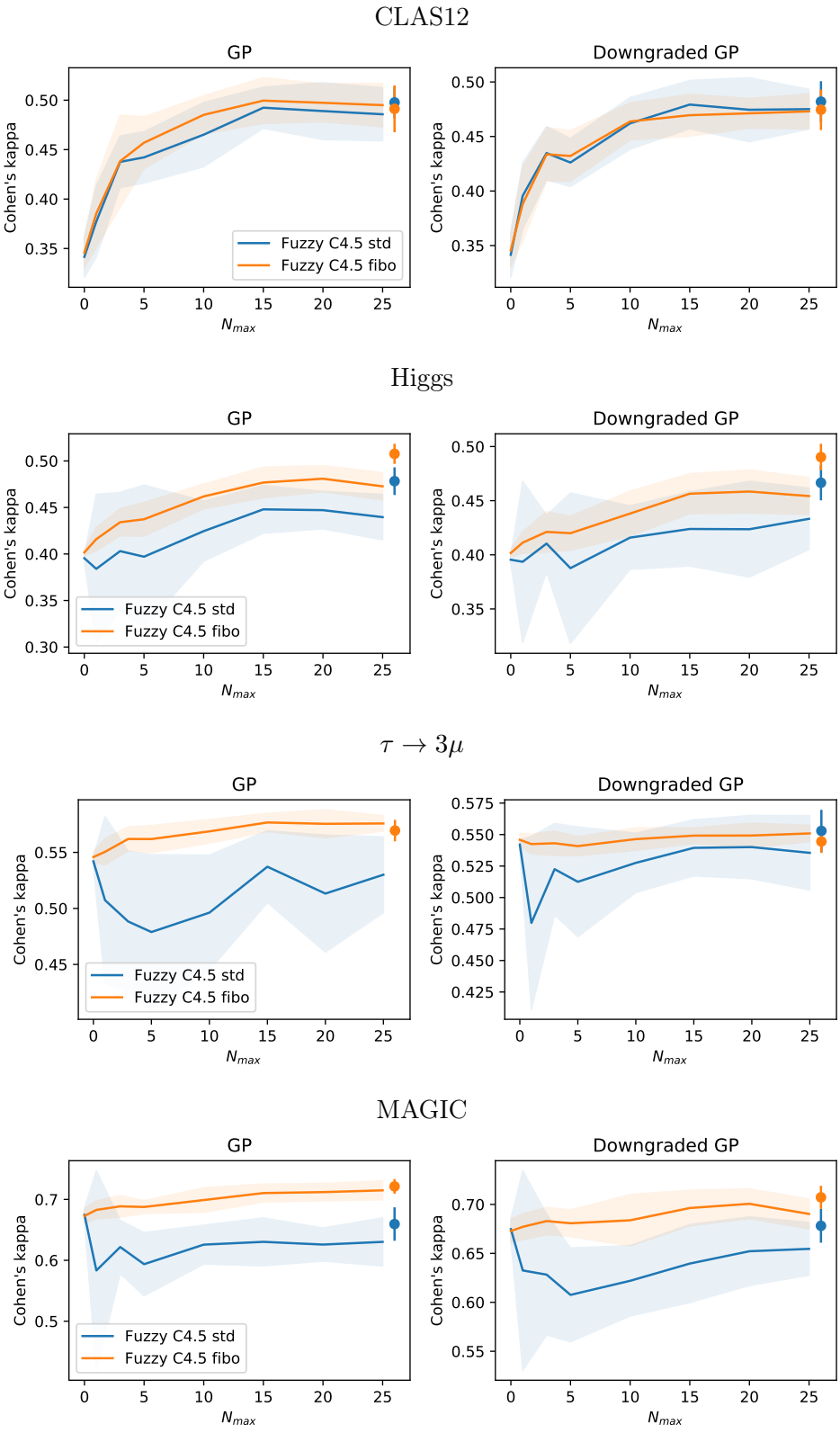


FIGURE E.1: Evolution of the Cohen's kappa score with the number of built features in fuzzy C4.5 std and Fibo. The markers on the right indicate the score obtained when building 100 features.

E.2 Fuzzy C4.5: Fibo version

TABLE E.2: Cohen's kappa score with different feature construction methods embedded into a fuzzy C4.5: Fibo version. The number of built features leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.346 ± 0.015	0.400 ± 0.003
MC	0.423 ± 0.018	0.453 ± 0.019
Downgraded GP	0.475 ± 0.019 (100)	0.490 ± 0.013 (100)
GP	**0.500 ± 0.023 (15)**	**0.508 ± 0.011 (100)**
	$\tau \rightarrow 3\mu$	MAGIC
Baseline (without feature construction)	0.545 ± 0.004	0.673 ± 0.012
MC	0.548 ± 0.008	0.703 ± 0.011
Downgraded GP	0.551 ± 0.006 (25)	0.707 ± 0.012 (100)
GP	**0.577 ± 0.008 (15)**	**0.721 ± 0.012 (100)**

E.3 CART

TABLE E.3: Cohen's kappa score with different feature construction methods embedded into CART. The number of built features leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.243 ± 0.016	0.276 ± 0.005
MC	0.281 ± 0.022	0.317 ± 0.016
Downgraded GP	0.368 ± 0.017 (25)	0.346 ± 0.081 (100)
GP	**0.437 ± 0.044 (100)**	**0.357 ± 0.036 (20)**
	$\tau \rightarrow 3\mu$	MAGIC
Baseline (without feature construction)	0.432 ± 0.007	0.598 ± 0.011
MC	0.444 ± 0.008	0.605 ± 0.016
Downgraded GP	0.459 ± 0.031 (25)	**0.634 ± 0.030 (100)**
GP	**0.481 ± 0.024 (100)**	0.627 ± 0.041 (20)

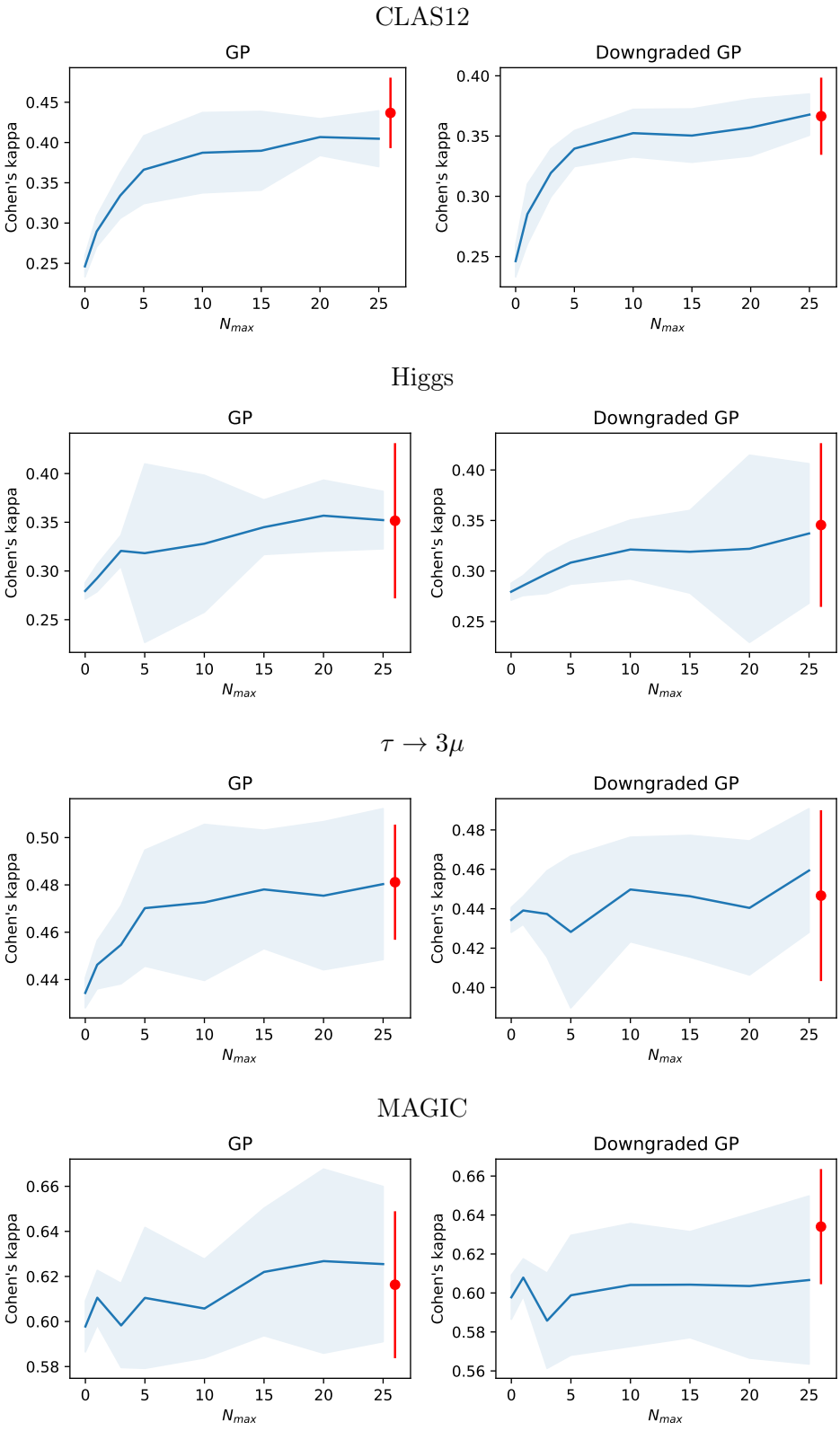


FIGURE E.2: Evolution of the Cohen's kappa score with the number of built features in CART. The red marker on the right indicates the score obtained when building 100 features.

E.4 AdaBoost

For Higgs and $\tau \to 3\mu$, no more than 50 features are built (1 per tree) for AdaBoost and GradientBoosting with the GP. The downgraded GP can build up to 350 features (7 per tree, i.e. the maximum) for all datasets.

TABLE E.4: Cohen's kappa score with different feature construction methods embedded into AdaBoost. The number of built features leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.333 ± 0.019	0.450 ± 0.014
MC	0.368 ± 0.011	0.483 ± 0.006
Downgraded GP	0.440 ± 0.014 (50)	**0.526 ± 0.008 (350)**
GP	**0.447 ± 0.015 (50)**	0.523 ± 0.008 (50)
	$\tau \to 3\mu$	MAGIC
Baseline (without feature construction)	0.643 ± 0.005	0.683 ± 0.012
MC	0.648 ± 0.006	0.700 ± 0.013
Downgraded GP	0.657 ± 0.007 (25)	0.720 ± 0.014 (5)
GP	**0.667 ± 0.007 (50)**	**0.729 ± 0.012 (50)**

E.5 GradientBoosting

TABLE E.5: Cohen's kappa score with different feature construction methods embedded into GradientBoosting. The number of built features leading to the best scores for GP and downgraded GP is specified in parentheses.

	CLAS12	Higgs
Baseline (without feature construction)	0.302 ± 0.012	0.387 ± 0.004
MC	0.407 ± 0.010	0.434 ± 0.005
Downgraded GP	**0.499 ± 0.010 (350)**	**0.467 ± 0.062 (350)**
GP	0.482 ± 0.032 (350)	0.394 ± 0.009 (5)
	$\tau \to 3\mu$	MAGIC
Baseline (without feature construction)	0.582 ± 0.006	0.686 ± 0.014
MC	**0.607 ± 0.006**	0.711 ± 0.015
Downgraded GP	0.598 ± 0.010 (50)	**0.713 ± 0.014 (350)**
GP	0.590 ± 0.009 (5)	0.685 ± 0.013 (3)

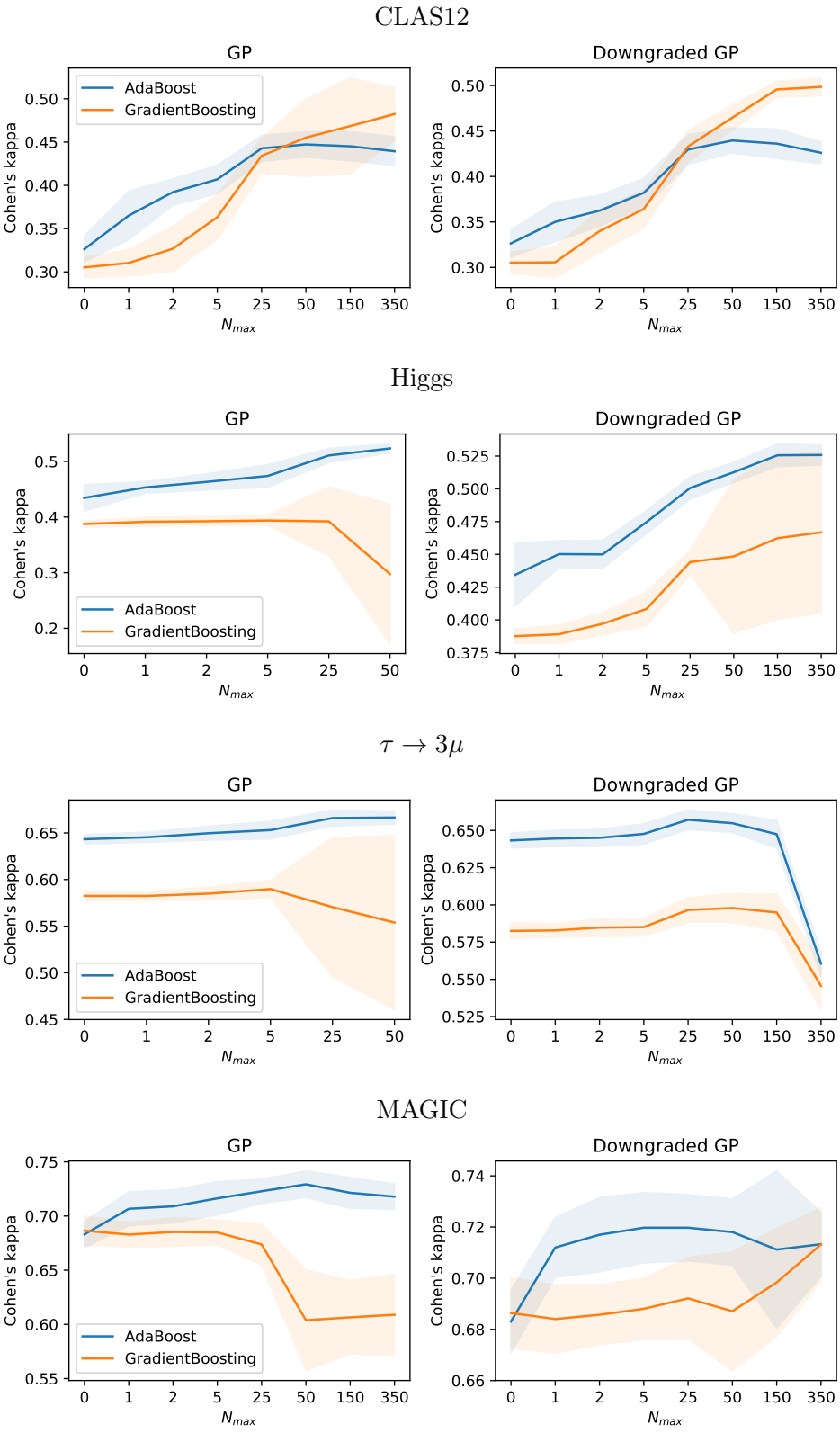


FIGURE E.3: Evolution of the Cohen's kappa score with the number of built features in AdaBoost and GradientBoosting.

Appendix F

Additional experiments on domain adaptation

F.1 Experiments with smeared simulated data with flat distributions

The training set comprises π^0 production events simulated with a flat distribution over the CLAS12 phase space. The validation set comprises 20% of such events, and the test set both DVCS and π^0 production events simulated with a flat distribution.

Hyperparameters have been tuned with the method detailed in 7.3.1:

- the regularization parameter of optimal transport is set to 0.1;
- FA MMD-net: 4 layers, batch size 512;
- FA WGAN-GP: 4 layers for both the generator and discriminator, batch size 256;
- additive MMD-net: 4 layers, batch size 512;
- additive WGAN-GP: 4 layers for both the generator and discriminator, batch size 512;
- additive FA MMD-net: 5 layers, batch size 512;
- additive FA WGAN-GP: 5 layers for both the generator and discriminator, batch size 512;
- additive random FA MMD-net: 5 layers, batch size 512;
- additive random FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 512.

Table F.1 presents the KS distances on the validation set.

Table F.2 presents the KS distances and area under the absolute difference of ROC curves (AUD) when the learnt mapping is applied on the test set generated with flat distributions.

The MMD-net seems to be a simpler model, whatever the variant, which does not necessarily need additional features to get good results. Indeed, the MMD distance used as loss function naturally considers the correlations between variables. On the opposite, the WGAN-GP seems to require more help to achieve its maximal performance, but obtains better results in the end.

TABLE F.1: Results on the validation set for flat distributions (π^0 production events only).

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}	$\min\left(r, \frac{1}{r}\right)$
Baseline	0.019	0.031	
Normalization	0.023	0.026	-0.80
CORAL	0.040	0.084	0.40
Optimal transport	0.017	0.026	0.40
FA MMD-net	0.097	0.052	0.27
FA WGAN-GP	0.105	0.040	0.11
Additive MMD-net	0.018	0.025	0.17
Additive WGAN-GP	0.019	0.031	0
Additive FA MMD-net	0.019	0.029	0
Additive FA WGAN-GP	0.008	0.021	0.91
Additive random FA MMD-net	0.019	0.027	0
Additive random FA WGAN-GP	**0.007**	**0.019**	0.92

F.2 Experiments with smeared simulated data with cross-sections

The training dataset consists of exclusive π^0 production events simulated with cross-sections. The validation set comprises 20% of such events. The test set is formed with DVCS and π^0 production events in equal amounts, simulated with cross-sections.

Hyperparameters with this training set are:

- the regularization parameter of optimal transport is set to 0.1;
- FA MMD-net: 4 layers, batch size 256;
- FA WGAN-GP: 8 layers for both the generator and discriminator, batch size 512;
- additive MMD-net: 4 layers, batch size 512;
- additive WGAN-GP: 6 layers for both the generator and discriminator, batch size 256;
- additive FA MMD-net: 6 layers, batch size 256;
- additive FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 256;
- additive random FA MMD-net: 4 layers, batch size 256;
- additive random FA WGAN-GP: 7 layers for both the generator and discriminator, batch size 256.

Table F.3 presents the KS distances on the validation set.

Table F.4 presents the KS distances and AUD metrics when the learnt mapping is applied on the test set generated with cross-sections.

TABLE F.2: Results on the test set generated with flat distributions (DVCS and π^0 production events).

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}
Baseline	0.104	0.047
Normalization	0.061	0.023
CORAL	0.050	0.090
Optimal transport	0.049	0.036
FA MMD-net	0.031	0.019
FA WGAN-GP	0.114	0.167
Additive MMD-net	0.047	0.021
Additive WGAN-GP	0.040	0.079
Additive FA MMD-net	0.062	0.024
Additive FA WGAN-GP	**0.016**	0.023
Additive random FA MMD-net	0.058	0.019
Additive random FA WGAN-GP	0.067	**0.015**
	$AUD_{\text{Fuzzy C4.5}}$	AUD_{FCGAM}
Baseline	0.070	0.043
Normalization	0.049	0.034
CORAL	0.022	0.010
Optimal transport	0.021	0.033
FA MMD-net	**0.008**	0.024
FA WGAN-GP	0.071	0.022
Additive MMD-net	0.040	0.033
Additive WGAN-GP	0.011	**0.006**
Additive FA MMD-net	0.048	0.036
Additive FA WGAN-GP	0.009	0.012
Additive random FA MMD-net	0.045	0.035
Additive random FA WGAN-GP	**0.008**	0.007

Table F.5 presents the KS distances and AUD metrics when the learnt mapping is applied on the reduced test set.

TABLE F.3: Results on the validation set for cross-sections distributions (π^0 production events only).

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}	$\min\left(r, \frac{1}{r}\right)$
Baseline	0.041	0.089	
Normalization	0.067	0.038	-0.51
CORAL	0.369	0.392	0.92
Optimal transport	0.046	0.087	-0.4
FA MMD-net	0.041	0.028	0
FA WGAN-GP	0.041	0.043	0
Additive MMD-net	0.035	0.032	0.11
Additive WGAN-GP	**0.013**	0.026	**0.44**
Additive FA MMD-net	0.017	0.028	0.39
Additive FA WGAN-GP	0.020	0.029	0.35
Additive random FA MMD-net	0.022	**0.017**	0.26
Additive random FA WGAN-GP	0.018	0.029	0.38

TABLE F.4: Results on the test set generated with cross-sections (DVCS and π^0 production events).

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}
Baseline	0.090	0.047
Normalization	0.046	0.185
CORAL	0.143	0.187
Optimal transport	0.062	0.057
FA MMD-net	0.040	**0.032**
FA WGAN-GP	0.113	0.206
Additive MMD-net	0.114	0.046
Additive WGAN-GP	0.172	0.131
Additive FA MMD-net	0.025	0.049
Additive FA WGAN-GP	0.138	0.141
Additive random FA MMD-net	**0.024**	0.036
Additive random FA WGAN-GP	0.096	0.058
	$AUD_{\text{Fuzzy C4.5}}$	AUD_{FCGAM}
Baseline	0.040	0.028
Normalization	0.017	0.009
CORAL	0.097	0.029
Optimal transport	0.036	0.049
FA MMD-net	0.023	0.014
FA WGAN-GP	0.114	0.056
Additive MMD-net	0.053	0.011
Additive WGAN-GP	0.100	0.054
Additive FA MMD-net	0.011	**0.009**
Additive FA WGAN-GP	0.085	0.056
Additive random FA MMD-net	**0.007**	0.010
Additive random FA WGAN-GP	0.040	0.014

TABLE F.5: Results on the reduced test set (DVCS and π^0 production events).

	$KS_{\text{Fuzzy C4.5}}$	KS_{FCGAM}
Baseline	0.080	0.043
Normalization	0.032	0.124
CORAL	0.167	0.213
Optimal transport	0.058	0.056
FA MMD-net	0.028	0.018
FA WGAN-GP	0.069	0.097
Additive MMD-net	0.040	0.037
Additive WGAN-GP	0.136	0.140
Additive FA MMD-net	0.024	0.044
Additive FA WGAN-GP	0.088	0.098
Additive random FA MMD-net	**0.015**	**0.014**
Additive random FA WGAN-GP	0.063	0.068
	$AUD_{\text{Fuzzy C4.5}}$	AUD_{FCGAM}
Baseline	0.038	0.035
Normalization	0.017	0.011
CORAL	0.095	0.035
Optimal transport	0.023	0.043
FA MMD-net	0.015	0.019
FA WGAN-GP	0.048	0.021
Additive MMD-net	**0.010**	**0.012**
Additive WGAN-GP	0.093	0.054
Additive FA MMD-net	0.012	0.015
Additive FA WGAN-GP	0.062	0.044
Additive random FA MMD-net	0.013	0.016
Additive random FA WGAN-GP	0.030	0.013

Appendix G

Complete interpretability survey and responses

The two parts of the interpretability survey discussed in chapter 8 are concatenated starting from next page. Then, the GAM model the FURIA model and the neural network referenced to as a link are displayed. The responses are displayed just after.

Interpretability survey

Thank you for accepting to answer this survey. The objective is to evaluate the interpretability of machine learning models developed at CEA Saclay for the analysis of CLAS12 data.

The survey is divided in two parts, sent separately by email. The first part of this questionnaire should take about 30 minutes to complete.
*Obligatoire

1. First, please provide us a "keyword" (for instance: your birth town, name of your dog, first name of your grandmother...) that will be used to correlate the two parts of the survey: *

_____

Using machine learning in experimental physics

2. Which degree of transparency would you estimate necessary for each of these tasks? *

Une seule réponse possible par ligne.

	No need for any explanation as long as the method is working well	The method should be sufficiently transparent to be validated	Complete understanding of the method is far more important than performance
Tracking	◯	◯	◯
Particle identification	◯	◯	◯
Event selection	◯	◯	◯
Smearing simulation to imitate data	◯	◯	◯

We would like to apply machine learning algorithms to isolate the deeply virtual Compton scattering events: an electron beam e interacts with a proton target p at rest. The final state is composed of the scattered electron e', the recoil proton p' as well as a multi-GeV gamma-photon.

$$e + p \rightarrow e' + p' + \gamma_1$$

The main source of background is the exclusive production of Pi0 which has the same final state as DVCS except that the photon is replaced by a Pi0-meson with an energy as high as the photon in DVCS. The latter immediately decays into two photons and its energy is randomly shared between them: Some decays produce photons with almost same energy while others produce a pair with a multi-GeV photon and only a few MeV photon.

$$e + p \rightarrow e' + p' + \pi^0 \rightarrow e' + p' + \gamma_1 + \gamma_2$$

Last but not least, the analysis considers events with a third photon gamma_3 in the final state, if any from background.

We developed an algorithm to build automatically high-level variables, such as a momentum balance or a missing mass, from the particle's three-momenta. Then, interpretable machine learning algorithms are trained using these high-level variables. The next sections have as objective to evaluate both the interpretability of these built variables and the interpretability of the machine learning models themselves.

Evaluation of high-level variables

Below are several high-level variables. The majority of them have been designed by a machine learning algorithm with varying settings. For each group, could you rate the presented variables according to your ability to understand what they mean and why they are considered discriminative for the event selection problem?

You will probably find that some of the presented variables are really really weird, it is normal: we compare different algorithms. The separating potential of the variables does not count here, only their understandability according to your perception.

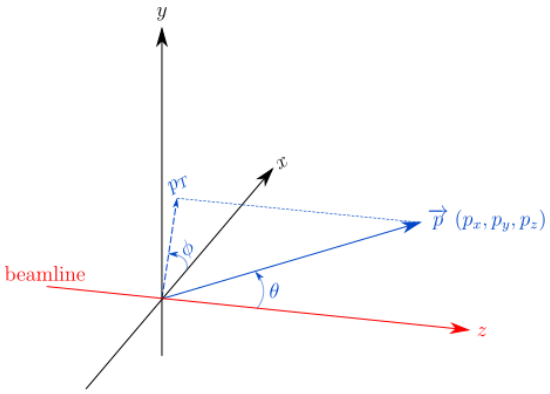
Base features from which the high-level variables are built. Energies and momenta are expressed in GeV and won't exceed 10.6 GeV (the incoming beam energy). Angles are expressed in radians. Up to three photons are considered (only one or two exist in case of DVCS or Pi0 events, but a third photon can come from background or from electron radiation). We rank the three photons from the most likely to be a DVCS event to the least likely, based on the missing mass ep->epγX.

$$\text{Electron: } \overrightarrow{p^e} = \begin{pmatrix} p_x^e \\ p_y^e \\ p_z^e \end{pmatrix} \text{ or } \begin{pmatrix} p_T^e \\ \theta^e \\ \phi^e \end{pmatrix}, \qquad \text{Proton: } \overrightarrow{p^p} = \begin{pmatrix} p_x^p \\ p_y^p \\ p_z^p \end{pmatrix} \text{ or } \begin{pmatrix} p_T^p \\ \theta^p \\ \phi^p \end{pmatrix},$$

$$\text{Photon number } i\text{: } \overrightarrow{p^{\gamma_i}} = \begin{pmatrix} p_x^{\gamma_i} \\ p_y^{\gamma_i} \\ p_z^{\gamma_i} \end{pmatrix} \text{ or } \begin{pmatrix} p_T^{\gamma_i} \\ \theta^{\gamma_i} \\ \phi^{\gamma_i} \end{pmatrix}$$

$$\text{Incoming beam: } \overrightarrow{e^{in}} = \begin{pmatrix} 0 \\ 0 \\ e_z^{in} \end{pmatrix}, \quad e_z^{in} = 10.6\,\text{GeV}$$

$$\text{Target proton: } \overrightarrow{p^{in}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad M_p = 0.938\ \text{GeV/c}^2$$



3. How would you rate these 8 variables, from poorly understandable (1) to highly understandable (5)? Do not hesitate to use the full range of ratings. *

Variable 1: $-\theta^{p} + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, 2\overrightarrow{p^{e}}\right) + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{p}}\right)$

Variable 2: $\text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_2}} + \overrightarrow{p^{\gamma_1}}\right)$

Variable 3: $\text{angle}\left(\overrightarrow{p^{\gamma_1}}, \overrightarrow{e^{in}} - \overrightarrow{p^{e}} - \overrightarrow{p^{p}}\right)$

Variable 4: $p_T^e + p_T^p + p_T^{\gamma_1}$

Variable 5: $p_z^p - e_z^{in} + p_z^e + p_z^{\gamma_1}$

Variable 6: $\overrightarrow{p^{\gamma_3}} \cdot \left(\overrightarrow{p^{e}} + \overrightarrow{p^{p}}\right)$

Variable 7: $\sqrt{\left(-\left\|\overrightarrow{p^{e}}\right\| - \left\|\overrightarrow{p^{\gamma_1}}\right\| + M_p + e_z^{in}\right)^2 - \left\|-\overrightarrow{p^{e}} - \overrightarrow{p^{\gamma_1}} + \overrightarrow{e^{in}}\right\|^2}$

Variable 8: $e_z^{in} + M_p - \left\|\overrightarrow{p^{e}}\right\| - \sqrt{\left\|\overrightarrow{p^{p}}\right\|^2 + M_p^2} - \left\|\overrightarrow{p^{\gamma_1}}\right\|$

Une seule réponse possible par ligne.

	Poorly understandable	2	3	4	Highly understandable
Variable 1	◯	◯	◯	◯	◯
Variable 2	◯	◯	◯	◯	◯
Variable 3	◯	◯	◯	◯	◯
Variable 4	◯	◯	◯	◯	◯
Variable 5	◯	◯	◯	◯	◯
Variable 6	◯	◯	◯	◯	◯
Variable 7	◯	◯	◯	◯	◯
Variable 8	◯	◯	◯	◯	◯

4. Do you have any additional comments?

5. How would you rate these 8 variables, from poorly understandable (1) to highly understandable (5)? Do not hesitate to use the full range of ratings. *

Variable 1: $\overrightarrow{p^{\gamma_3}} \cdot \left(\overrightarrow{p^e} + \overrightarrow{p^p}\right)$

Variable 2: $\frac{-\text{angle}\left(\overrightarrow{p^{\gamma_1}}, \overrightarrow{p^{\gamma_2}}\right)}{\theta^{\gamma_1} + \theta^p}$

Variable 3: $\text{angle}\left(\overrightarrow{p^e}, \overrightarrow{p^p} + \overrightarrow{p^{\gamma_1}} - \overrightarrow{e^{in}}\right)$

Variable 4: $\left(2\overrightarrow{p^{\gamma_1}} - \overrightarrow{p^{\gamma_2}}\right) \cdot \left(\overrightarrow{p^{\gamma_1}} + \overrightarrow{p^p} + \overrightarrow{p^e} - \overrightarrow{e^{in}}\right)$

Variable 5: $\left\|\overrightarrow{p^{\gamma_1}} - \overrightarrow{e^{in}} + \overrightarrow{p^p} + \overrightarrow{p^e}\right\|$

Variable 6: $-\theta^p + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, 2\overrightarrow{p^e}\right) + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^p}\right)$

Variable 7: $\text{angle}\left(-\overrightarrow{p^{\gamma_2}}, 2\overrightarrow{p^{\gamma_1}} + \overrightarrow{p^{\gamma_2}}\right)$

Variable 8: $-\overrightarrow{p^{\gamma_1}} \cdot \left(\overrightarrow{p^e} + \overrightarrow{p^p}\right) + p_x^p p_y^{\gamma_1}$

Une seule réponse possible par ligne.

	Poorly understandable	2	3	4	Highly understandable
Variable 1	◯	◯	◯	◯	◯
Variable 2	◯	◯	◯	◯	◯
Variable 3	◯	◯	◯	◯	◯
Variable 4	◯	◯	◯	◯	◯
Variable 5	◯	◯	◯	◯	◯
Variable 6	◯	◯	◯	◯	◯
Variable 7	◯	◯	◯	◯	◯
Variable 8	◯	◯	◯	◯	◯

6. Do you have any additional comments?

Evaluation of a parametric model

This part aims at evaluating the interpretability of a model called a GAM (generalized additive model). A classification GAM produces a single output ŷ between 0 and 1 to classify events, 0 being signal and 1 contamination. The model consists in a sum of univariate smooth functions as displayed below.

$$\hat{y} = \frac{1}{1 + e^{-F(X)}}$$

$$F(X) = \beta_0 + f_1(x_1) + f_2(x_2) + ... + f_D(x_D)$$

The sum F(X) is linked to the output ŷ by a sigmoid function: the higher the value of F(X), the closer ŷ gets to 1, and conversely. The variables x_i can be momenta components or high-level variables such as an energy balance. The f_i functions are smooth functions determined throughout a training on simulation data.

At this link: https://tinyurl.com/y47fuygl (link to a PDF file) is a GAM trained on CLAS12 simulation data. It comprises 5 terms. The higher the return value, the higher the probability to have a background event. Conversely, the closest the return value is to 0, the higher the probability to have a DVCS event. We placed on the plots the values that will be useful to answer next question.

7. For instance, how would an event with the following characteristics be classified by the GAM (i.e. ŷ is closer to 1 (Pi0) or 0 (DVCS))? (Necessary high-level variables computed from these values are printed directly on the GAM plots in the PDF file) *

$$\text{Electron: } \vec{p^e} = \begin{pmatrix} -0.3 \\ 1.0 \\ 5.0 \end{pmatrix}, \qquad \text{Proton: } \vec{p^p} = \begin{pmatrix} 0.6 \\ 0.2 \\ 2.8 \end{pmatrix},$$

$$\text{Photon number 1: } \vec{p^{\gamma_1}} = \begin{pmatrix} -0.3 \\ -1.3 \\ 2.5 \end{pmatrix},$$

$$\text{Photon number 2: } \vec{p^{\gamma_2}} = \begin{pmatrix} -0.2 \\ -0.4 \\ 0.8 \end{pmatrix},$$

$$\text{Photon number 3: } \vec{p^{\gamma_3}} = \begin{pmatrix} 0.2 \\ 0.4 \\ -0.6 \end{pmatrix}$$

Une seule réponse possible.

- ◯ Signal event (DVCS) (i.e. ŷ close to 0)
- ◯ Background event (Pi0) (i.e. ŷ close to 1)
- ◯ I don't know

8. Which term would you delete in order for this specific example to be classified differently? *

Une seule réponse possible.

- ◯ Term 1 (f1(x1))
- ◯ Term 2 (f2(x2))
- ◯ Term 3 (f3(x3))
- ◯ Term 4 (f4(x4))
- ◯ Term 5 (f5(x5))
- ◯ I don't know

9. How confident are you in your answers? *

Une seule réponse possible.

- ◯ That was a random guess
- ◯ 25% sure
- ◯ 50% sure
- ◯ 75% sure
- ◯ 100% sure

10. Do you think this model is sufficiently transparent for the analysis to be easily validated? By validation, we mean a retro-engineering of the decision process regarding the classification of events to understand the performances of the algorithm in a given region of the phase space. *

Une seule réponse possible.

	1	2	3	4	5	
Not at all, I cannot imagine how to validate such a model	◯	◯	◯	◯	◯	Completely transparent, easy validation

11. Do you have any comments on this model?

Interpretability survey

Thank you for accepting to answer the second part of this survey.

As a reminder, the objective is to evaluate the interpretability of machine learning models developed at CEA Saclay for the analysis of CLAS12 data.

This second part of the questionnaire should take about 30 minutes to complete.
*Obligatoire

1. First, could you remind us your "keyword" that you entered in the first part of the survey? (for instance: your birth town, name of your dog, first name of your grandmother...) *

As a reminder, we would like to apply machine learning algorithms to isolate the deeply virtual Compton scattering events: an electron beam e interacts with a proton target p at rest. The final state is composed of the scattered electron e', the recoil proton p' as well as a multi-GeV gamma-photon.

$$e + p \rightarrow e' + p' + \gamma_1$$

The main source of background is the exclusive production of Pi0 which has the same final state as DVCS except that the photon is replaced by a Pi0-meson with an energy as high as the photon in DVCS. The latter immediately decays into two photons and its energy is randomly shared between them: Some decays produce photons with almost same energy while others produce a pair with a multi-GeV photon and only a few MeV photon.

$$e + p \rightarrow e' + p' + \pi^0 \rightarrow e' + p' + \gamma_1 + \gamma_2$$

Last but not least, the analysis considers events with a third photon gamma_3 in the final state, if any from background.

We developed an algorithm to build automatically high-level variables, such as a momentum balance or a missing mass, from the particle's three-momenta. Then, interpretable machine learning algorithms are trained using these high-level variables. The next sections have as objective to evaluate both the interpretability of these built variables and the interpretability of the machine learning models themselves.

Evaluation of high-level variables

Below are several high-level variables. The majority of them have been designed by a machine learning algorithm with varying settings. For each group, could you rate the presented variables according to your ability to understand what they mean and why they are considered discriminative for the event selection problem?

You will probably find that some of the presented variables are really really weird, it is normal: we compare different algorithms. The separating potential of the variables does not count here, only their understandability according to your perception.

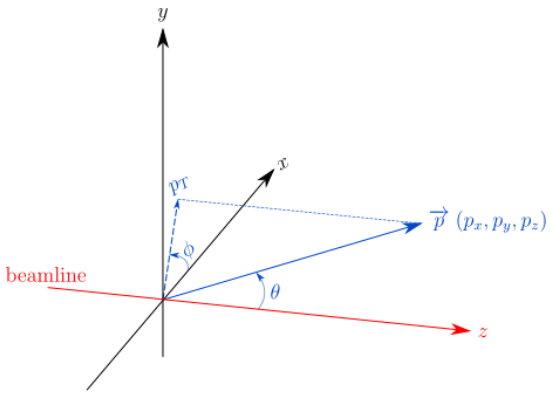
Base features from which the high-level variables are built. Energies and momenta are expressed in GeV and won't exceed 10.6 GeV (the incoming beam energy). Angles are expressed in radians. Up to three photons are considered (only one or two exist in case of DVCS or Pi0 events, but a third photon can come from background or from electron radiation). We rank the three photons from the most likely to be a DVCS event to the least likely, based on the missing mass ep->epγX.

$$\text{Electron: } \overrightarrow{p^e} = \begin{pmatrix} p_x^e \\ p_y^e \\ p_z^e \end{pmatrix} \text{ or } \begin{pmatrix} p_T^e \\ \theta^e \\ \phi^e \end{pmatrix}, \qquad \text{Proton: } \overrightarrow{p^p} = \begin{pmatrix} p_x^p \\ p_y^p \\ p_z^p \end{pmatrix} \text{ or } \begin{pmatrix} p_T^p \\ \theta^p \\ \phi^p \end{pmatrix},$$

$$\text{Photon number } i\text{: } \overrightarrow{p^{\gamma_i}} = \begin{pmatrix} p_x^{\gamma_i} \\ p_y^{\gamma_i} \\ p_z^{\gamma_i} \end{pmatrix} \text{ or } \begin{pmatrix} p_T^{\gamma_i} \\ \theta^{\gamma_i} \\ \phi^{\gamma_i} \end{pmatrix}$$

$$\text{Incoming beam: } \overrightarrow{e^{in}} = \begin{pmatrix} 0 \\ 0 \\ e_z^{in} \end{pmatrix}, \quad e_z^{in} = 10.6\,\text{GeV}$$

$$\text{Target proton: } \overrightarrow{p^{in}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad M_p = 0.938\ \text{GeV/c}^2$$



2. How would you rate these 8 variables, from poorly understandable (1) to highly understandable (5)? Do not hesitate to use the full range of ratings. *

Variable 1: $\cos\left(p_z^e + \sqrt{p_T^{\gamma_2}} + p_z^{\gamma_1} + p_z^p\right)$

Variable 2: $\text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_1}}\right) + \text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_2}} + \overrightarrow{p^{\gamma_1}}\right)$

Variable 3: $\left(\overrightarrow{p^p} + \overrightarrow{p^{\gamma_1}}\right) \cdot \left(\overrightarrow{e^{in}} - \overrightarrow{p^{\gamma_1}} - \overrightarrow{p^p} - \overrightarrow{p^e}\right)$

Variable 4: $p_T^e \left(p_T^{\gamma_1} + p_T^p\right)^2 + \tan\left(p_z^{\gamma_2}\right)$

Variable 5: $\sqrt{p_z^{\gamma_1} + p_z^p + p_z^e \cos\left(p_z^p\right)}$

Variable 6: $p_z^e + p_z^p + p_z^{\gamma_1} + p_T^{\gamma_2} - p_y^{\gamma_3} \sin\left(\cos\left(p_z^{\gamma_3}\right)\right)$

Variable 7: $p_z^p - e_z^{in} + p_z^e + p_z^{\gamma_1}$

Variable 8: $\left(p_T^p + \left\|\overrightarrow{p^{\gamma_2}}\right\|\right) \tan\left(\text{angle}\left(\overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_1}}\right)\right)$

Une seule réponse possible par ligne.

	Poorly understandable	2	3	4	Highly understandable
Variable 1	◯	◯	◯	◯	◯
Variable 2	◯	◯	◯	◯	◯
Variable 3	◯	◯	◯	◯	◯
Variable 4	◯	◯	◯	◯	◯
Variable 5	◯	◯	◯	◯	◯
Variable 6	◯	◯	◯	◯	◯
Variable 7	◯	◯	◯	◯	◯
Variable 8	◯	◯	◯	◯	◯

3. Do you have any additional comments?

Evaluation of a rule-based model

This part aims at evaluating the interpretability of a rule-based model called FURIA. A rule base consists of a set of rules. More specifically, a FURIA rule base consists in two parts, one for each class (DVCS, Pi0). To evaluate how a given example is classified, one must consider the two parts separately. For each part (i.e. class), one must add the confidence values of each rule that is validated by the example (i.e. the rule covers the given example). Therefore, a "score" is obtained for each class being the sum of the confidence values of the triggered rules. The predicted class is the one obtaining the maximal score.

At this link: https://tinyurl.com/y6a8wof9 (link to a PDF file) is a FURIA model trained on CLAS12 simulation data. It comprises 15 rules optimized on training data. We placed under the rules the values that will be useful to answer next question.

4. For instance, how would an event with the following characteristics be classified by the FURIA model? (Necessary high-level variables computed from these values are printed directly under the FURIA rules in the PDF file) *

$$\text{Electron: } \vec{p^{\,e}} = \begin{pmatrix} -0.3 \\ 1.0 \\ 5.0 \end{pmatrix}, \qquad \text{Proton: } \vec{p^{\,p}} = \begin{pmatrix} 0.6 \\ 0.2 \\ 2.8 \end{pmatrix},$$

$$\text{Photon number 1: } \vec{p^{\,\gamma_1}} = \begin{pmatrix} -0.3 \\ -1.3 \\ 2.5 \end{pmatrix},$$

$$\text{Photon number 2: } \vec{p^{\,\gamma_2}} = \begin{pmatrix} -0.2 \\ -0.4 \\ 0.8 \end{pmatrix},$$

$$\text{Photon number 3: } \vec{p^{\,\gamma_3}} = \begin{pmatrix} 0.2 \\ 0.4 \\ -0.6 \end{pmatrix}$$

Une seule réponse possible.

- ◯ Signal event (DVCS)
- ◯ Background event (Pi0)
- ◯ I don't know

5. Which rule would you delete in order for this event to be classified differently? *

Une seule réponse possible.

- ◯ Rule 1
- ◯ Rule 2
- ◯ Rule 3
- ◯ Rule 4
- ◯ Rule 5
- ◯ Rule 6
- ◯ Rule 7
- ◯ Rule 8
- ◯ Rule 9
- ◯ Rule 10
- ◯ Rule 11
- ◯ Rule 12
- ◯ Rule 13
- ◯ Rule 14
- ◯ Rule 15
- ◯ I don't know

6. How confident are you in your answers? *

Une seule réponse possible.

- ◯ That was a random guess
- ◯ 25% sure
- ◯ 50% sure
- ◯ 75% sure
- ◯ 100% sure

7. Do you think this model is sufficiently transparent for the analysis to be easily validated? By validation, we mean a retro-engineering of the decision process regarding the classification of events to understand the performances of the algorithm in a given region of the phase space. *

Une seule réponse possible.

	1	2	3	4	5	
Not at all, I cannot imagine how to validate such a model	◯	◯	◯	◯	◯	Completely transparent, easy validation

8. Do you have any comments on this model?

Evaluation of a neural network

This part aims now at evaluating a neural network. A neural network consists in non-linear operations on the inputs.

A network has been trained on CLAS12 simulation data and is available at this link: https://tinyurl.com/y2sbps64 (link to a PDF file). It comprises 1281 weights in total that have been optimized on training data. From these weights are extracted the relative importances of the different input variables (displayed on the right side in the PDF). This reflects the variables that impact the most the classification depending on their values (color from blue to red). A negative SHAP value means a negative effect on the output (more likely a Pi0 event), while a positive SHAP value pushes towards the output being a DVCS event.

No other information is available about the network. Next to the feature importance study are values that will be useful for the next question.

9. For instance, how would an event with the following characteristics be classified by the neural network? (Necessary variables are printed directly on the right side of the PDF file, along with corresponding colors) *

$$\text{Electron: } \vec{p^{\,e}} = \begin{pmatrix} -0.3 \\ 1.0 \\ 5.0 \end{pmatrix}, \qquad \text{Proton: } \vec{p^{\,p}} = \begin{pmatrix} 0.6 \\ 0.2 \\ 2.8 \end{pmatrix},$$

$$\text{Photon number 1: } \vec{p^{\,\gamma_1}} = \begin{pmatrix} -0.3 \\ -1.3 \\ 2.5 \end{pmatrix},$$

$$\text{Photon number 2: } \vec{p^{\,\gamma_2}} = \begin{pmatrix} -0.2 \\ -0.4 \\ 0.8 \end{pmatrix},$$

$$\text{Photon number 3: } \vec{p^{\,\gamma_3}} = \begin{pmatrix} 0.2 \\ 0.4 \\ -0.6 \end{pmatrix}$$

Une seule réponse possible.

- ◯ Signal event (DVCS)
- ◯ Background event (Pi0)
- ◯ I don't know

10. How confident are you in your answer? *

Une seule réponse possible.

- ◯ That was a random guess
- ◯ 25% sure
- ◯ 50% sure
- ◯ 75% sure
- ◯ 100% sure

11. Do you think this model is sufficiently transparent for the analysis to be easily validated? By validation, we mean a retro-engineering of the decision process regarding the classification of events to understand the performances of the algorithm in a given region of the phase space. *

Une seule réponse possible.

	1	2	3	4	5	
Not at all, I cannot imagine how to validate such a model	◯	◯	◯	◯	◯	Completely transparent, easy validation

12. Do you have any comments on this model?

Comparison between different models

In this part, you will be asked to subjectively compare four methods for event selection. The first three are the GAM model, the FURIA model and the neural network of the previous parts.

Model A: GAM

Available at this link: https://tinyurl.com/y47fuyql (link to a PDF file)

Model B: FURIA

Available at this link: https://tinyurl.com/y6a8wof9 (link to a PDF file)

Model C: neural network

Available at this link: https://tinyurl.com/y2sbps64 (link to a PDF file)

Model D: cuts

The fourth model is a list of cuts that a physicist would use for event selection:

An event is a DVCS event if:

$$\sqrt{\left(-\left\|\vec{p^{\prime}}\right\|-\left\|\vec{p^{\gamma_1}}\right\|+M_p+e_z^{in}\right)^2-\left\|-\vec{p^{\prime}}-\vec{p^{\gamma_1}}+\vec{e^{in}}\right\|^2}\in[0.1\,\text{GeV},1.7\,\text{GeV}]$$

$$\text{and angle}\left(\vec{p^{\gamma_1}},\vec{e^{in}}-\vec{p^{\prime}}-\vec{p^{p}}\right)\leq 0.6^{\circ}$$

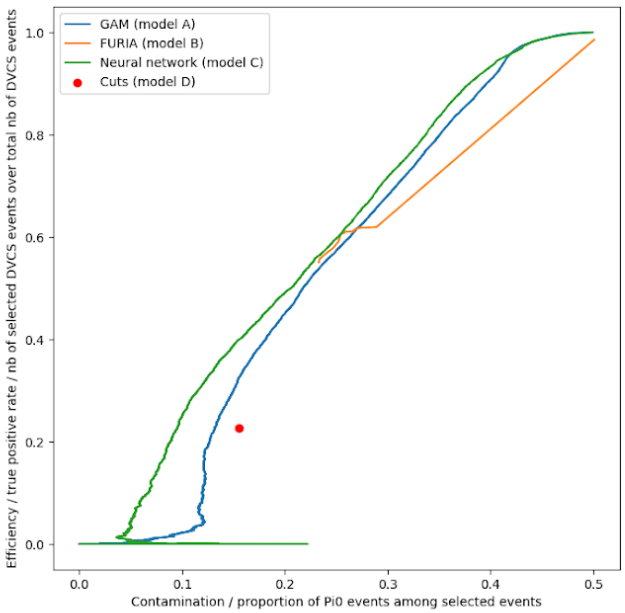
$$\text{and}\ \sqrt{\left(p_x^e+p_x^p+p_x^{\gamma_1}\right)^2+\left(p_y^e+p_y^p+p_y^{\gamma_1}\right)^2}\leq 0.12\,\text{GeV}$$

$$\text{and}\left(-\left\|\vec{p^{p}}\right\|-\sqrt{\left\|\vec{p^{p}}\right\|^2+M_p^2}-\left\|\vec{p^{\gamma_1}}\right\|+M_p+e_z^{in}\right)^2-\left\|-\vec{p^{\prime}}-\vec{p^{p}}-\vec{p^{\gamma_1}}+\vec{e^{in}}\right\|^2\in\left[-0.04\,\text{GeV}^2,0.04\,\text{GeV}^2\right]$$

$$\text{and}\ e_z^{in}+M_p-\left\|\vec{p^{\prime}}\right\|-\sqrt{\left\|\vec{p^{p}}\right\|^2+M_p^2}-\left\|\vec{p^{\gamma_1}}\right\|\in[-0.5\,\text{GeV},1.2\,\text{GeV}]$$

$$\text{and angle}\left(\vec{p^{\prime}},\vec{p^{\gamma_1}}\right)>10^{\circ}$$

Below is a chart representing the performances of the different models on a simulation dataset. In y-axis is the selection efficiency (i.e. the proportion of DVCS events actually retained by the event selection process). In x-axis is the Pi0 contamination (i.e. proportion of Pi0 events in the selected subset of events). Note that the machine learning models permit to fine-tune a posteriori the balance between the statistic and the contamination. However, FURIA has a limited flexibility.
Model D (cuts) retrieved 23% of signal events and its selected sample comprises 16% of Pi0 contamination.
Finally, it should be noted that a Pi0 subtraction technique should be able to remove the remaining contamination provided it does not exceed around 30% of the selected sample.



13. Now, please imagine for each model how you would validate it: try to predict how each model would perform on real data with different noise characteristics (different cross sections or additional sources) and/or different resolutions compared to the simulation used for training. Considering on the one hand the robustness and ease of validation of the different models, and on the other hand the performances displayed above, which model would you recommend to a PhD student or a post-doctoral fellow to perform an analysis? By this question, we would like to evaluate your personal tradeoff between performance and complexity to elaborate an associated validation procedure. *

Une seule réponse possible par ligne.

	The model I would prefer to use	Second model	Third model	The model I dislike the most
Model A (GAM)	◯	◯	◯	◯
Model B (FURIA)	◯	◯	◯	◯
Model C (neural network)	◯	◯	◯	◯
Model D (cuts)	◯	◯	◯	◯

14. Do you have any additional comments?

Questions about you

15. You are...

Une seule réponse possible.

- ◯ PhD student
- ◯ Post-doc
- ◯ Staff scientist
- ◯ Professor
- ◯ Autre : _____

16. What is your age?

Une seule réponse possible.

- ◯ 24 or less
- ◯ 25 to 34
- ◯ 35 to 49
- ◯ 50 to 64
- ◯ 65 or more

17. What is (are) your field(s) of expertise?

Plusieurs réponses possibles.

- ☐ Nuclear physics
- ☐ Hadronic physics
- ☐ Particle physics

18. How knowledgeable are you about machine learning/artificial intelligence?

Une seule réponse possible.

- ◯ I am a frequent user of machine learning techniques
- ◯ I am curious of these techniques or had once a student who used these techniques
- ◯ I do not know anything about machine learning

19. What are your expectations of applying machine learning techniques in physics?

Une seule réponse possible.

- ◯ I do not expect machine learning to change dramatically the reach of our experiments
- ◯ It can mostly improve the low-level analysis: tracking efficiency, particle identification, shower reconstruction, ...
- ◯ Having the full reconstruction-analysis chain with machine learning tools will improve significantly the physics output

20. How often do you practice programming?

Une seule réponse possible.

- ◯ Daily
- ◯ When there is a complex task too long to be done by a graduate student
- ◯ I no longer practice programming

21. What is your preferred programming language?

_____

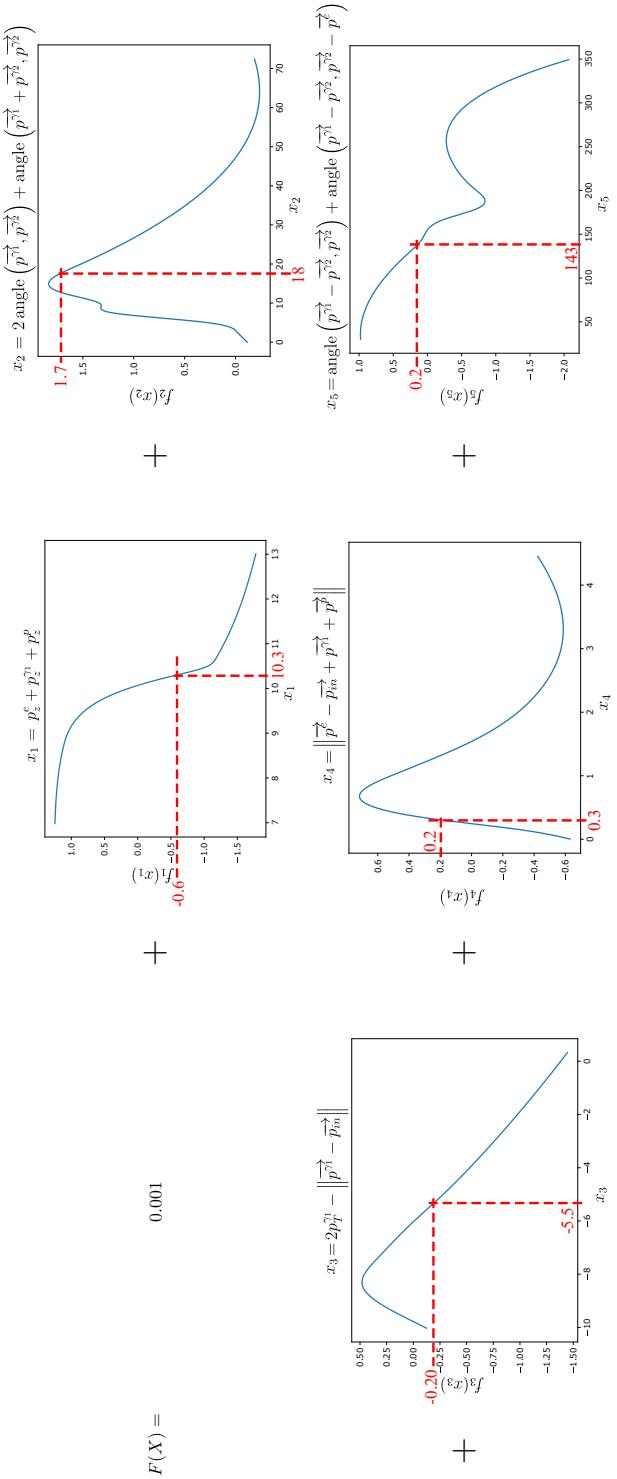
22. Thank you for answering this survey! You can leave us your email if you are willing to deepen the discussion later.

_____

Ce contenu n'est ni rédigé, ni cautionné par Google.

Google Forms

GAM


$F(X) =$
0.001
$f_1(x_1)$
$x_1 = p_z^g + p_z^{\gamma_1} + p_z^p$
-0.6
10.3
$f_2(x_2)$
$x_2 = 2\,\text{angle}\left(\overrightarrow{p^{\gamma_1}}, \overrightarrow{p^{\gamma_2}}\right) + \text{angle}\left(\overrightarrow{p^{\gamma_1}} + \overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_2}}\right)$
1.7
18
$f_3(x_3)$
$x_3 = 2p_T^{\gamma_1} - \left\|\overrightarrow{p^{\gamma_1}} - \overrightarrow{p_m}\right\|$
-0.20
-5.5
$f_4(x_4)$
$x_4 = \left\|\overrightarrow{p^g} - \overrightarrow{p_m} + \overrightarrow{p^{\gamma_1}} + \overrightarrow{p^p}\right\|$
0.2
0.3
$f_5(x_5)$
$x_5 = \text{angle}\left(\overrightarrow{p^{\gamma_1}} - \overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_2}}\right) + \text{angle}\left(\overrightarrow{p^{\gamma_1}} - \overrightarrow{p^{\gamma_2}}, \overrightarrow{p^{\gamma_2}} - \overrightarrow{p^g}\right)$
0.2
143

FURIA base

DVCS rules

Rule 1: If $\underset{\text{-0.3}}{p_z^p + p_z^e + p_z^{\gamma_1} - e_z^{in}} \geq -0.42$ and $\underset{\text{2.5}}{p_z^{\gamma_1}} \geq 1.64$ and $\underset{\text{106}}{\left\|\vec{p^p} + \vec{p^{\gamma_1}} + \vec{p^e}\right\|^2} \geq 109$ then DVCS ($c = 0.91$)

Rule 2: If $\underset{\text{106}}{\left\|\vec{p^p} + \vec{p^{\gamma_1}} + \vec{p^e}\right\|^2} \geq 104$ then DVCS ($c = 0.77$)

Rule 3: If $\underset{\text{1.3}}{p_T^{\gamma_1}} \geq 1.7$ then DVCS ($c = 0.89$)

Rule 4: If $\underset{\text{2.5}}{p_z^{\gamma_1}} \geq 4.3$ then DVCS ($c = 0.77$)

Rule 5: If $\underset{\text{1.3}}{p_T^{\gamma_1}} \geq 1.3$ and $\underset{\text{28}}{\theta^{\gamma_1}} \geq 26.2$ then DVCS ($c = 0.81$)

Pi0 rules

Rule 6: If $\underset{\text{106}}{\left\|\vec{p^p} + \vec{p^{\gamma_1}} + \vec{p^e}\right\|^2} \leq 104$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \leq 13$ and $\underset{\text{2.5}}{p_z^{\gamma_1}} \geq 1.3$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \geq 1.3$ then Pi0 ($c = 0.97$)

Rule 7: If $\underset{\text{-0.3}}{p_z^p + p_z^e + p_z^{\gamma_1} - e_z^{in}} \leq -0.42$ and $\underset{\text{1.0}}{p_T^e} \geq 1.4$ then Pi0 ($c = 0.76$)

Rule 8: If $\underset{\text{1.3}}{p_T^{\gamma_1}} \leq 0.27$ and $\underset{\text{106}}{\left\|\vec{p^p} + \vec{p^{\gamma_1}} + \vec{p^e}\right\|^2} \leq 99$ and $\underset{\text{13}}{\theta^p} \geq 36$ then Pi0 ($c = 0.89$)

Rule 9: If $\underset{\text{106}}{\left\|\vec{p^p} + \vec{p^{\gamma_1}} + \vec{p^e}\right\|^2} \leq 104$ and $\underset{\text{2.5}}{p_z^{\gamma_1}} \leq 4.2$ and $\underset{\text{108}}{\text{angle}\left(\vec{p^{\gamma_1}} - \vec{p^p}, \vec{p^e}\right)} \leq 128$ then Pi0 ($c = 0.77$)

Rule 10: If $\underset{\text{2.5}}{p_z^{\gamma_1}} \leq 2.3$ and $\underset{\text{107}}{\phi^e} \geq 36$ and $\underset{\text{1.0}}{p_y^e} \leq 0.97$ then Pi0 ($c = 0.66$)

Rule 11: If $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \leq 4.3$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \geq 1.19$ and $\underset{\text{0.8}}{p_z^{\gamma_2}} \geq 0.76$ and $\underset{\text{1.3}}{p_T^{\gamma_1}} \geq 0.30$ then Pi0 ($c = 0.96$)

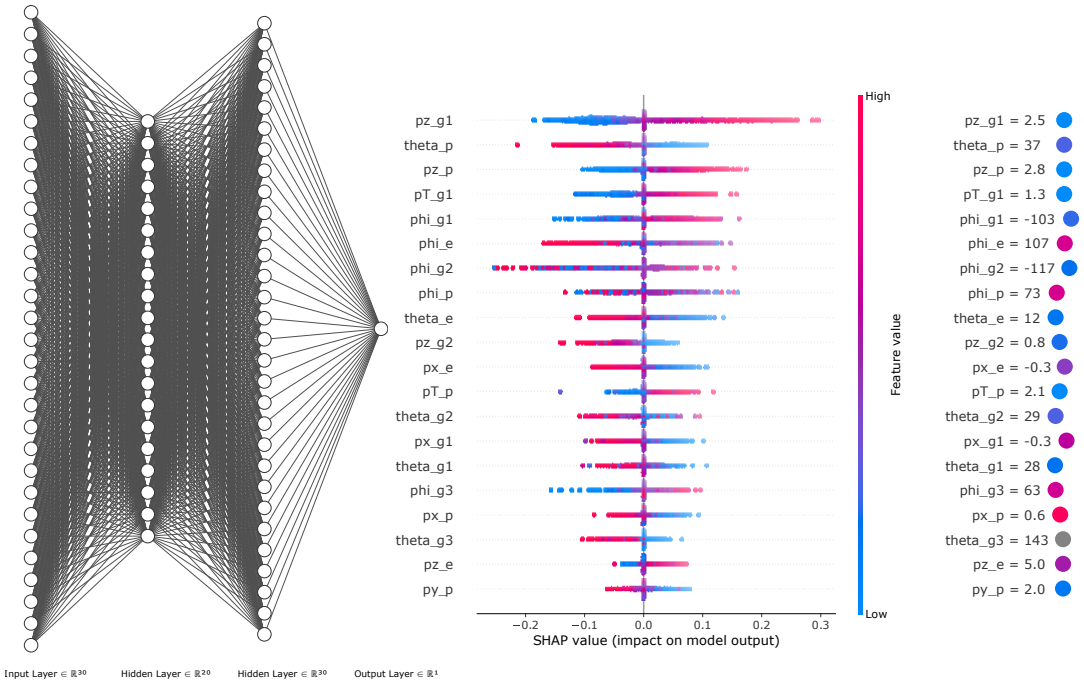
Rule 12: If $\underset{\text{2.5}}{p_z^{\gamma_1}} \leq 1.5$ and $\underset{\text{107}}{\phi^e} \geq -63$ and $\underset{\text{-0.3}}{p_x^{\gamma_1}} \leq -0.25$ and $\underset{\text{-0.3}}{p_x^e} \leq 1.5$ then Pi0 ($c = 0.76$)

Rule 13: If $\underset{\text{2.5}}{p_z^{\gamma_1}} \leq 2.6$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \leq 23$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \geq 2.9$ and $\underset{\text{0.2}}{p_x^{\gamma_2}} \leq 0.25$ then Pi0 ($c = 0.81$)

Rule 14: If $\underset{\text{1.3}}{p_T^{\gamma_1}} \leq 0.98$ and $\underset{\text{2.5}}{p_z^{\gamma_1}} \leq 2.3$ and $\underset{\text{0.6}}{p_T^p} \leq 1.5$ and $\underset{\text{108}}{\text{angle}\left(\vec{p^{\gamma_1}} - \vec{p^p}, \vec{p^e}\right)} \leq 150$ and $\underset{\text{1.0}}{p_T^e} \leq 1.4$ then Pi0 ($c = 0.73$)

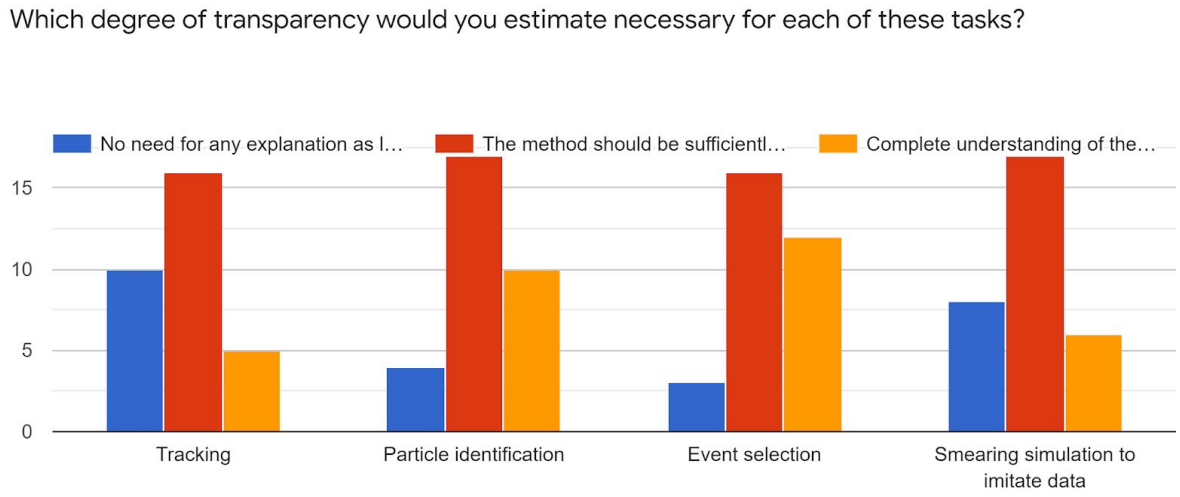
Rule 15: If $\underset{\text{1.3}}{p_T^{\gamma_1}} \leq 0.83$ and $\underset{\text{0.8}}{p_z^{\gamma_2}} \geq 0.83$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \geq 1.1$ and $\underset{\text{5.0}}{\text{angle}\left(\vec{p^{\gamma_2}}, \vec{p^{\gamma_1}} + \vec{p^{\gamma_2}}\right)} \leq 4.7$ then Pi0 ($c = 0.9$)

Neural network

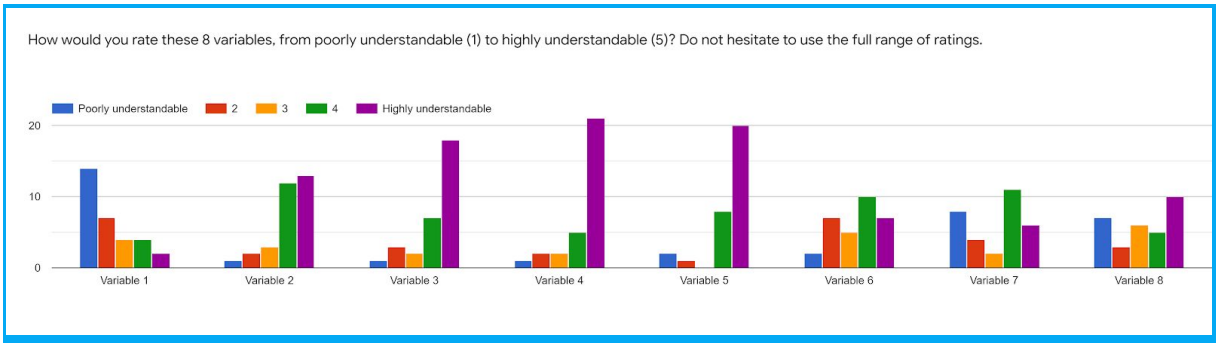


Part 1

Using machine learning in experimental physics



Evaluation of high-level variables



Do you have any additional comments?

10 answers

Consider recasting the variables in invariant form for applicability to EIC

I would think invariant mass of two photons would be useful too in distinguishing pi0 vs DVCS.

No

These variable has some physics meaning

May be with oral explanations I could understand more variables

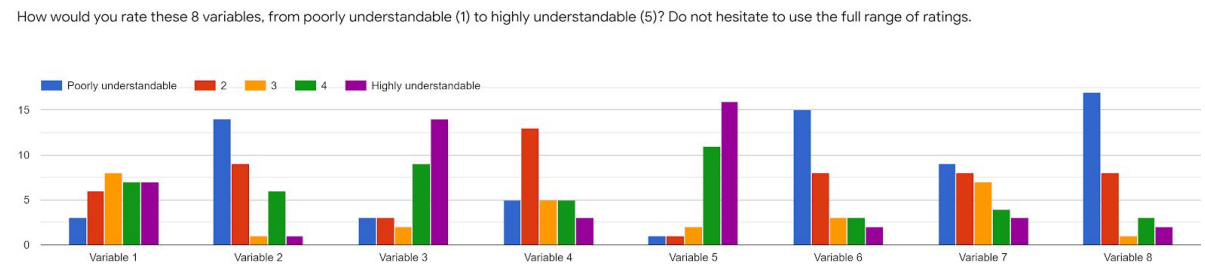
In variable 1, what is the meaning of the angle between TWICE the momentum vector of the electron and one of the gammas?

What is the gamma_3?

I can't quite understand why the first variable's middle term is angle(p^gamma2, 2*p^e). Is the coefficient of p^2, "2", necessary?

It is highly unclear how the reader's vector algebra proficiency is relevant at all in how the machine learning algorithm is supposed to work. I always thought that the promise of ML is to let the computer find out features and combinations thereof that might escape a human. The fact that a variable makes or does not make sense or has an immediate physical interpretation (again, for humans) should not be a reason for preventing a ML algorithm from using it. That said this reviewer is highly skeptical of variables that combine several/many individually measured features, especially as there could/are correlations between these that might, or might not be fully taken into account and/or might be underestimated.

The understandability seems almost binary to me: either I understand what the variable is, and then it's absolutely clear, or I kind of see what the expression is doing but I don't see exactly why.



Do you have any additional comments?

6 answers

in Variable 6, when calculating angle why instead of p_e, the 2p_e is used? the result should be the same right?

No

those variable has no physics meaning at all, just a mathematical variable

Why we need such variables? of course, we can invent whatever we want. If you explain the purpose of these variables, I can answer. Without this, I am completely confused

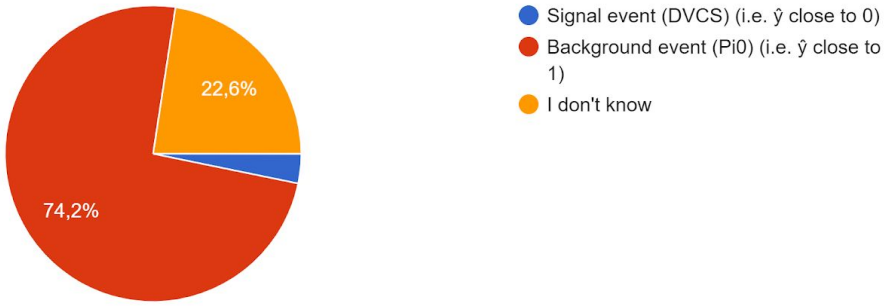
See above comment. One would also want to add that QE (quantification of error) in ML is relatively straightforward only in linear regression cases AND when only random uncertainties are present. Most of the work envisioned here does not fall in that category.

Variable 1 here is the same as variable 6 in the block above (and variable 1 is the same as variable 6 in the block above).

Evaluation of a parametric model

For instance, how would an event with the following characteristics be classified by the GAM (i.e. ŷ is closer to 1 (Pi0) or 0 (DVCS))? (Necessary high-l...e printed directly on the GAM plots in the PDF file)

31 réponses

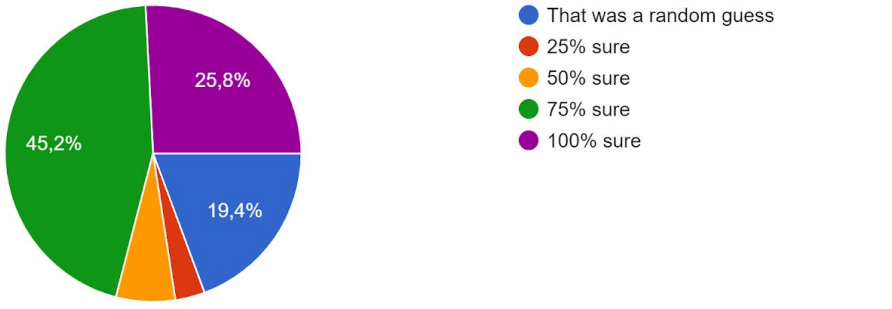


Which term would you delete in order for this specific example to be classified differently?

31 réponses

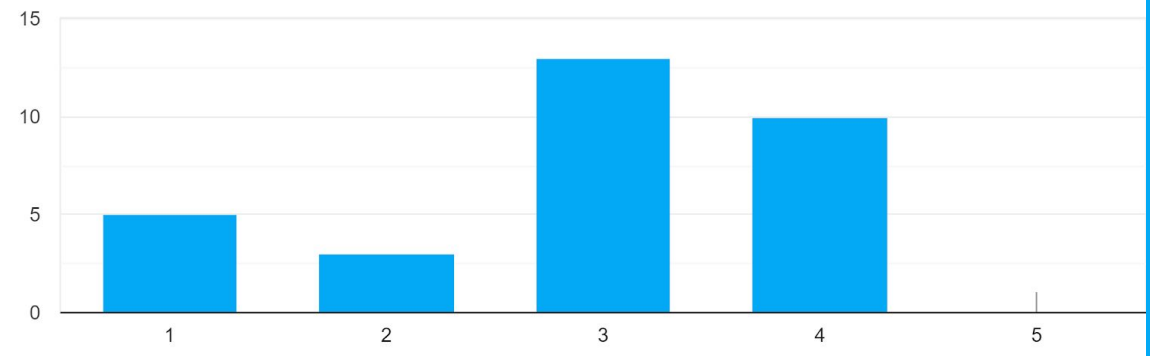
How confident are you in your answers?

31 réponses



Do you think this model is sufficiently transparent for the analysis to be easily validated? By validation, we mean a retro-engineering of the dec...e algorithm in a given region of the phase space.

31 réponses



Do you have any comments on this model?

11 answers

I think that the correlation between the variables Xi should be studied- I believe that this is already done. It is hardly possible to validate the model without undertanding the correlation between the variables, understanding the variable is improtant and not essential, but in separating signal/background, it is improtenter to understand correlations between different variables

pi0 vs dvcs separation is one of the most straight-forward aspects of DVMP analysis. Harder topics are tracking and pid. Are you using this as a test case, since we have confidence in the "classic" approach, or is this intended as a major advance in DVCS analysis?

How the functions f1(x1), f2(x2) etc are normalized?, i.e do they have a certain integral?

in some x_i there is p_in in the formulae, but if I am not mistaken, p_in is zero, therefore redundant in the formulae (and slightly misleading).

I was expecting that the x_i were a subset of the variables shown in the previous page

too little information

X5 seems to be strange; not sure if it adds sensitivity.

I was not able to understand it completely.

il aurait été bon de rappeler l'energie du faisceau pour aider et la notation dans le fichier GAM est incorrecte (vect(p_in) est en fait vect(e_in)).

The most discriminant term (f2) is one of the less "explanable" to me...
GAMs do not produce really transparent results even though they are optimal, wrt to a certain training.

The danger to me would be the bias introduced by the training of this model which makes some variable more discriminant than others for a specific training set. This can be called a systematic error...

This seems to me less trustworthy than a classical analysis where this bias is more controlled, or at least understandable.

Moreover how to put the frontier between a signal and background event ? Is 0.5 the best choice ? What is the shape of the probability distribution for a set of known signal events, (and background events), is the separation completely clear (or figure of merit ?)

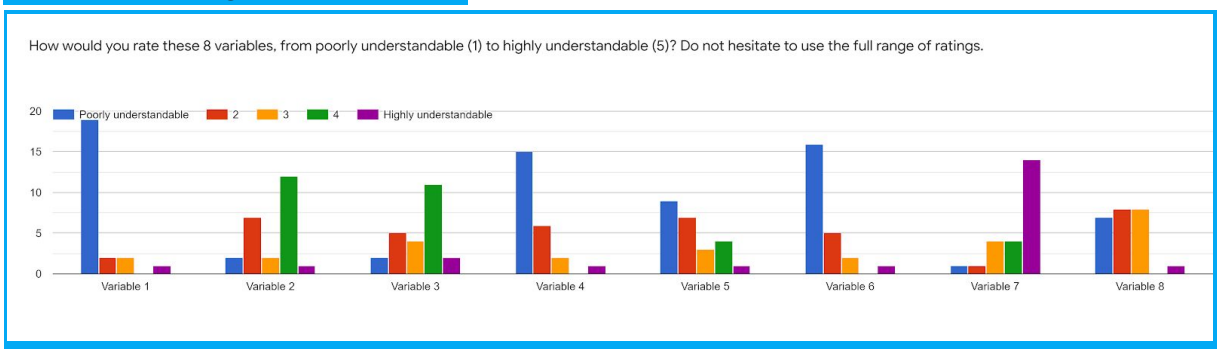
While this reviewer appreciates the intent of this survey, the way the questions are formulated is somewhat problematic. The algorithm described at the top of this page is a regression (summing f1, f2, ..., plugging into the chosen activation function, sigmoid in this case (why not tanh? softsign? or any other choice). Then the reviewer is asked to turn this regression into a classification and pass judgement whether the data points to a DVCS or a background event. While it is straightforward to find out that the 1.7 term is the one that influences the most the output of the sigmoid (and thus its presence/absence will modify the regression output the most) LACKING a threshold value for the classification means that the output is PURE guesswork. One would (wrongfully) assume that the threshold would be halfway between 0 and 1 (so 0.5) thus possibly changing the classification by removing the second term. However, that threshold might as well be 0.9 (again, we are not told!), in which case the second does not make a difference in the classification. Thus, the model is not completely described and asking reviewers to pass judgement on it is not very constructive.

Not really, but this exercise was fun! And an excellent way of explaining how this ML method works

---

Part 2

Evaluation of high-level variables



Do you have any additional comments?

8 answers

the expressions are clear, the physical meaning is not clear sometimes, but I can understand the idea that those variables might have a good separation power based on whether the event is signal or background!

What is the meaning of a cosine of a momentum?

trigonometric functions acting on non-pure numbers are meaningless

No

Not familiar enough to judge the understanding of the variables.

Why you are asking 90% of your questions? What do you want?

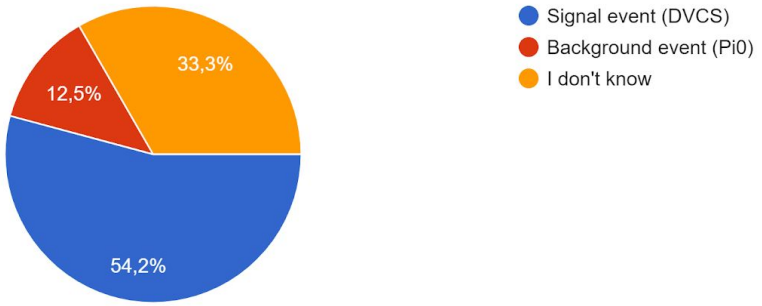
"Understandable" and "interpret-able" from the physics standpoint are, in this reviewer's mind, two totally different terms. The fact that one knows how to manipulate elements of an abstract algebra (like, say, 4 vectors), does not mean that any/all operations one can carry out with said objects have direct/simple physics interpretation. As noted earlier, this (in)ability of interpreting particular combinations should not deter one from using them in a ML program.

I think I'm missing something -- all of these seem really obscure, except the simple momentum balance. Also, cosine of a momentum? That seems odd...

Evaluation of a rule-based model

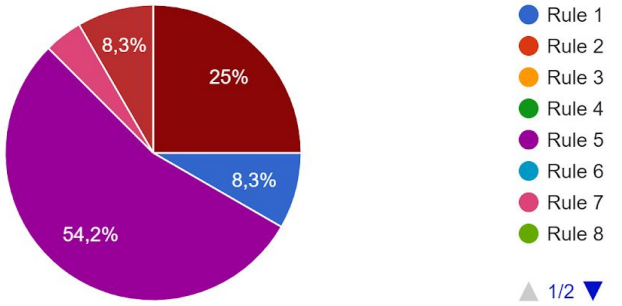
For instance, how would an event with the following characteristics be classified by the FURIA model? (Necessary high-level variables computed fr...ed directly under the FURIA rules in the PDF file)

24 réponses



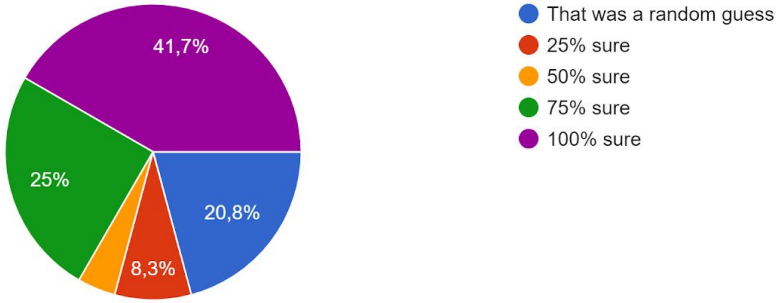
Which rule would you delete in order for this event to be classified differently?

24 réponses



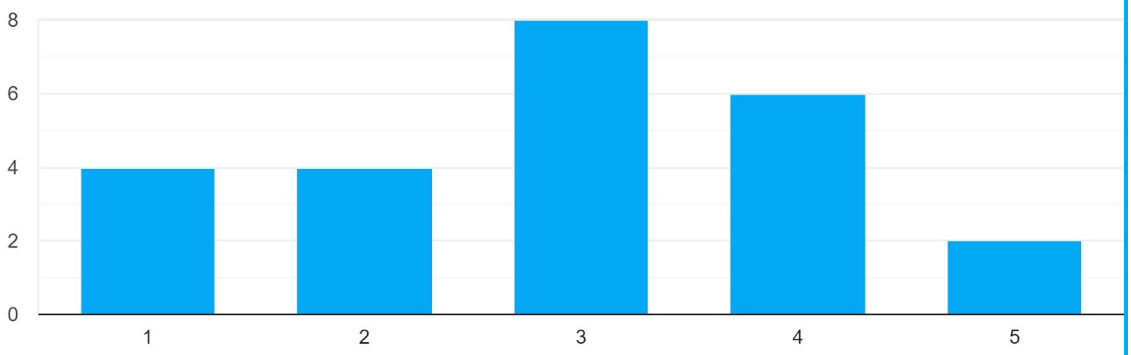
How confident are you in your answers?

24 réponses



Do you think this model is sufficiently transparent for the analysis to be easily validated? By validation, we mean a retro-engineering of the dec...e algorithm in a given region of the phase space.

24 réponses



Do you have any comments on this model?

7 answers

it is transparent, it is clear that the set of rules are built from training the algorithm, however, as before, I believe that the thing that is missing to validate the model is a study of the correlation betweenthe variables, using two variables that are highly correlated in your algorithm would bias the outcome !

I presume the c-values are confidence values, but no information was given on how the c-values are computed. The event scored equally on the dvcs and pi0 scales so I have no idea how to use, let alone validate the model.

No

More black box like than the first model

Poorly explained model

1) How are determined the numbered values indicated in the condition of the rules ? (and the score ?)
Would it be interesting to have distributions instead of raw numbers ? (a smooth real valued function (score) taking the z-momentum balance as parameter ?)

2) How are considered the correlations between rules ? Can one simply sums the output of 2 correlated rules ?
Would it be interesting to have a smooth score function taking all the variables and constructing a score out of them ?

3) The conditions inside one rule can be correlated, and to me, can be called "cuts". Is this conceptually really better than a usual "cut'and'count" analysis ?

4) I guess the main goal is to find the structure of the signal (or background) inside a phase space region, and try to discriminate it via a simple real valued function (score)
(To integrate non linear correlation laying inside data, would it be interesting to apply a manifold learning algorithm to understand the structure of signal for instance, and put a notion of distance for one event to this manifold structure ? but would it be interpretable....)

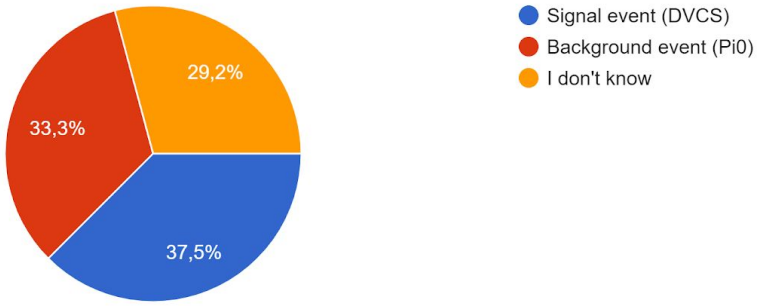
5) Why only 5 rules for signal and 10 for pi0 ?

interesting point: "rules" are easily understood and with clear interpretation, their combination is a bit more involved if done "by hand" (though a program that knows how to do joint probability is relatively easy to implement).

Evaluation of a neural network

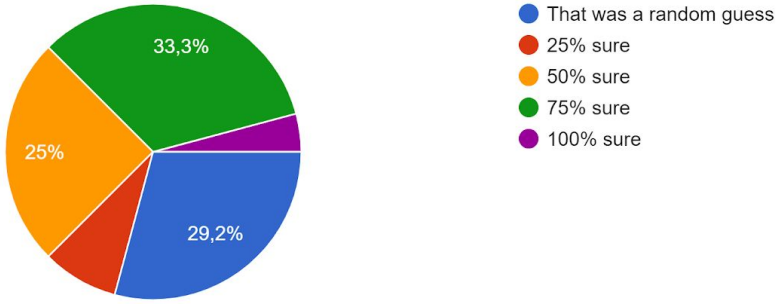
For instance, how would an event with the following characteristics be classified by the neural network? (Necessary variables are printed directly ... of the PDF file, along with corresponding colors)

24 réponses



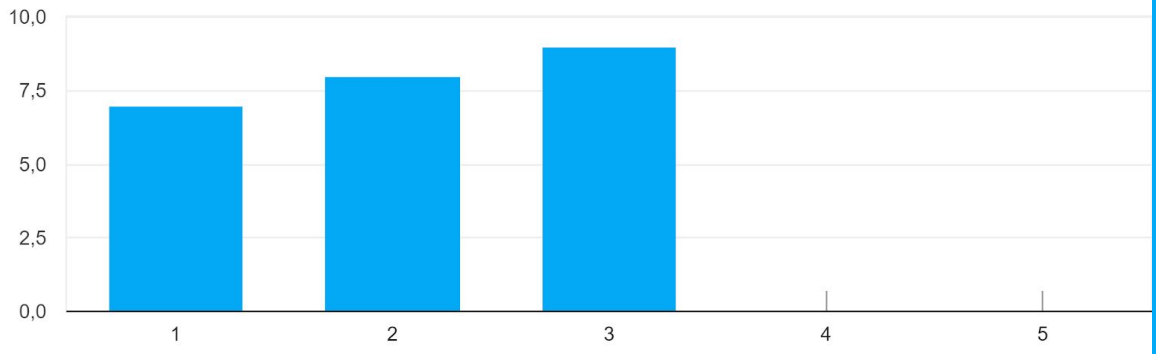
How confident are you in your answer?

24 réponses



Do you think this model is sufficiently transparent for the analysis to be easily validated? By validation, we mean a retro-engineering of the dec...e algorithm in a given region of the phase space.

24 réponses



Do you have any comments on this model?

8 answers

It is not clear what it means "relative importance"

Why theta_g3 is grey ?

No

Not enough explication given

I can not understand why you conduct such survey

With the file MAROUEN, maybe the event is more blue than red, but VERY DIFFICULT to conclude. You cannot convince colleagues without presenting efficiency and purity and changing the initial sample (degrading resolution or adding other type of background).

I did not get the model ; To me the pdf seems really ill-posed...what is a shap value ?

In the pdf, next to "shap value" is written "impact on model output" , this is ambiguous and can be understood as the importance of a variable in the classification (but it is not, this is for the color I guess ?...)

So one should look on the variables with red dots because they are more important in the classification. OK

Now why do we need the values of the observables ? is it the shap value ? should we sum them weighted with the color ?

I guess then most of the red variables are positive, so positive shap ? dvcs event ?

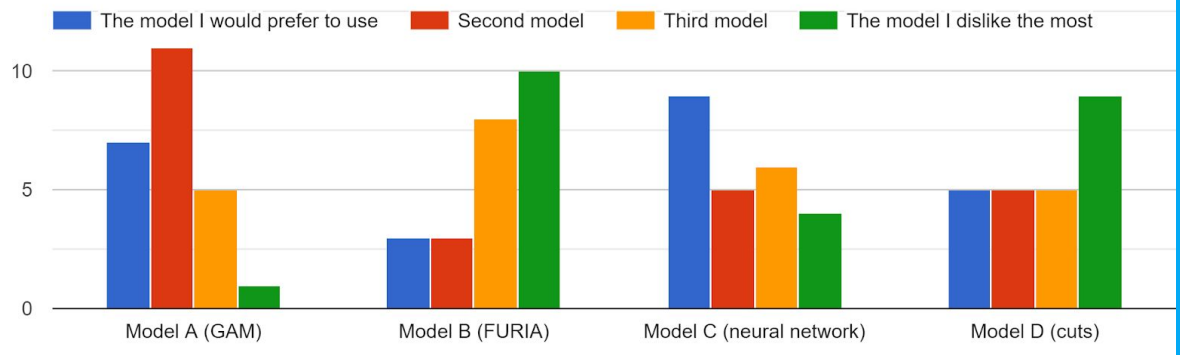
I would tend not to trust a model I do not understand the output :)

“When I use a word, it means just what I choose it to mean — neither more nor less." (Lewis Carroll). If one is interested in obtaining precise answers then one should start by asking well formulated questions: In this case there are clearly NOT 1281 "weights" as the question implies but rather 1200 weights and 81 thresholds or biases. Furthermore, the absolute values of the input features is not that informative as one does (should anyway!) need to scale the input variables. That scaling (min/max, normalization, other?) is not specified. Not at all clear what the middle panel in the plot is supposed to represent. One suspects that it is the distribution of each feature (normalized maybe?). Is this the distribution of the training or of the testing sample (or both, or neither?). A more informative set of numbers would have been the actual weights of the synapses between neurons (thickness of the line in the LHS plot, maybe?).

I'm not totally sure that I'm interpreting the information correctly. Also, the colours are not enough to tell how far along the SHAP scale each variable is. But it does give an overall feel..

Comparison between different models

Now, please imagine for each model how you would validate it: try to predict how each model would perform on real data with different noise ch...y to elaborate an associated validation procedure.



Do you have any additional comments?

11 answers

I think that there is a bias in the study, comparing different models should be when te models are trained using the same input variables, it seems that for the NN you used a different set of variables; same goes for the other two models. so how to tell whether the degradation in the performance or the imrpvement is due to choice of variables or choice of technology?

The photon angle cut (last cut in model D) seems more of a kinematic cut (exclude regions of high BH contribution, low CLAS12 acceptance, and high background) rather than a signal vs pi0 rejection. Therefore I think it is rejecting events that have nothing to do with dvcs vs pi0 and is biasing the comparison with ML. I ranked the Model B worst, because there seems no chance of comparing it with Model D. Also, by your criterion of pi0 subtraction by MC, Model B is unusable.

In order to validate any of A, B, or C, model D will be developed to some extent on simulated events. Thus, model D will be available to apply and to validate whichever of A, B, or C is chosen. It says that you can accommodate up to 30% pion contamination. That seems to indicate you can obtain approx. 2.5 times the number of events using A, B, or C, or a reduction of about 1.5 in the statistical uncertainty. The heart of this question is whether or not A, B, or C is introducing a systematic bias of that size or greater? To some extent Model D is used to estimate this, and some extent, the ability to validate the procedure by "understanding" the variables on which models A, B, or C use does this.

For the "cuts" model you give only one point, i.e. a specific set of cuts. But in principle, one can study the signal and background fraction when varying the cuts.

I rate worst FURIA, because it looks the less easy to train and run

No

The NN is the best working model, but 1) it's a completely black box 2) I didn't understand your explanation 3) people not familiar with them, dislike them

Sorry, I can not see any sense of such survey

To be convince by a NN I would like to see the impact on efficiency and purity if the sample is changed (this is life, we do not know the starting case!)

To me, all the methods are far more complicated to implement than a usual "cuts" analysis. Furia provides interesting improvements to be considered.

Although the neural network is the most efficient I cannot imagine its behaviour when introducing noise and smearing... They can be perfect for other tasks than data selection though...

Moreover, the data selection is not sufficient by itself. At the end of the extraction of some physics (cross section I guess in this case), the acceptances plays a major role in the correction on the resulting events ; namely it corrects for all the geometrical leakage of detection with respect to the MC generation by computing the ratio "detected&selected" / generated. Acceptances correct many aspects of an analysis, from geometrical inefficiencies to reconstruction errors/bias, and of course event selection. The question would be what is the improvement in the acceptance when using a classical "cuts" analysis or machine learning techniques (neural networks or furia, or gam, or whatever) ?

A validation procedure can be a comparison with the litterature on a previously measured cross section for instance. As long as the errors are well evaluated (and they depend on acceptances !), why not using even a neural network if they provide consistent results and better precision....

Not particularly clear what is represented on the x axis. It is pretty clear (hopefully) that the vertical represents the proportion of the DVCS events a particular model finds. The horizontal, however, is not clear: does a value of 0.1 on the horizontal means that, on top of good DVCS events the model mis-identified pi0 events which amount to 10% of the selected sample or a value of 0.1 means that the model picked up 10% of the pi0 events in the original simulation? In either case it would have been way more informative if one were told what is the ratio of DVCS to pi0 events in the simulation. For Model D it would have been informative to vary the size of the cuts to get a dependence rather than a single point. One suspects that with very generous cuts one will pick up all DVCS events while not incurring (at least not according to the plot) more than

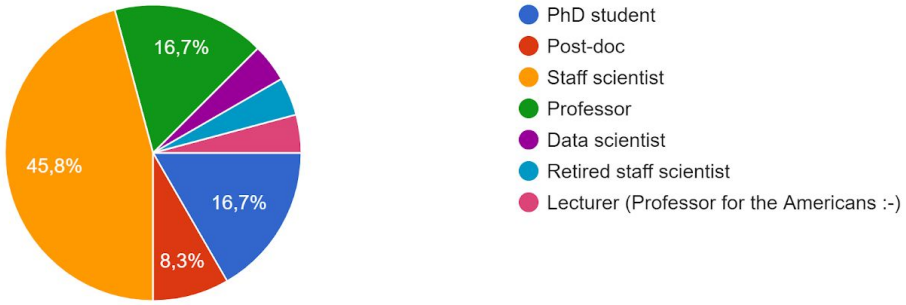
50% pi0 background. Last note: statistical error on a background subtracted distribution is not equal to the uncertainty of a pure sample of the same size.

GAM is easy to understand and very transparent and seems robust. It performs second best in the region just below 30% of pi0 contamination, where you maximise your DVCS signal but can still subtract pi0. FURIA is also easy to understand, but seems to be a set of fancy probability-weighted cuts and though it performs slightly better for lower pi0 contamination, it seems to be only applicable to just above 20%. I feel somehow more uneasy relying on it. Neutral Networks clearly show the best performance, but, based purely on the info in this survey, I don't feel I am confident enough understanding how to use the information.This can be fixed with a more in-depth study, though, so I might change my preference. For a PhD student or a postdoc just starting out, GAM might be a clearer starting point. Good old-fashioned cuts suck, of course.

Questions about you

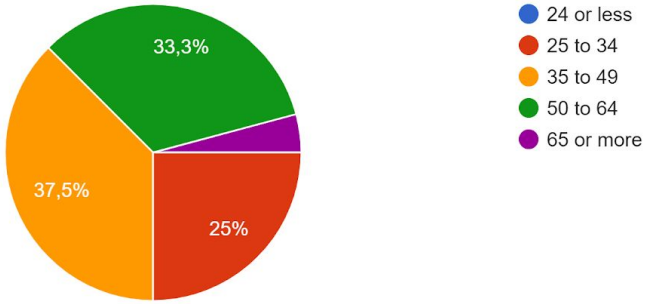
You are...

24 réponses



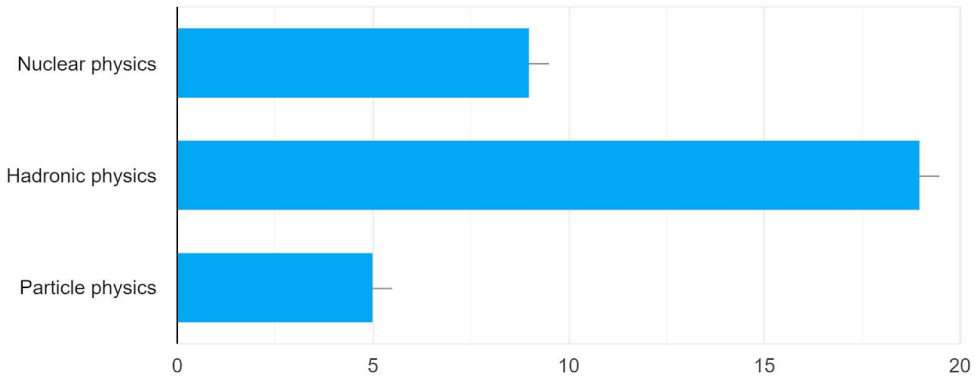
What is your age?

24 réponses



What is (are) your field(s) of expertise?

23 réponses



How knowledgeable are you about machine learning/artificial intelligence?

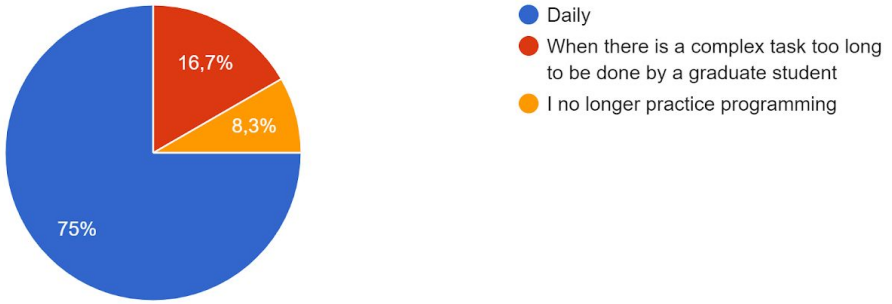
24 réponses

What are your expectations of applying machine learning techniques in physics?

24 réponses

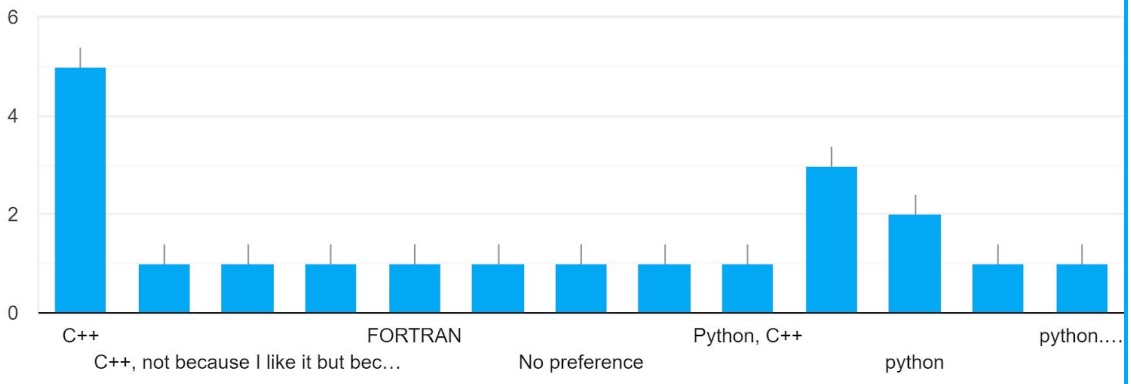
How often do you practice programming?

24 réponses



What is your preferred programming language?

20 réponses



Thank you for answering this survey! You can leave us your email if you are willing to deepen the discussion later.

8 answers

(We prefer not to share the emails of the respondents that agreed to provide them to us)

Appendix H

Asymmetries using additional models

This appendix lists the asymmetry plots obtained with different event selection techniques:

- fuzzy C4.5 with momentum corrections;
- fuzzy C4.5 of depth 3 with momentum corrections;
- FCGAM with momentum corrections;
- FCGAM of 5 terms with momentum corrections;
- FURIA with momentum corrections;
- cuts with momentum corrections;
- neural network with momentum corrections;
- fuzzy C4.5 without momentum corrections;
- FCGAM without momentum corrections;
- FURIA without momentum corrections;
- transferred fuzzy C4.5;
- retrained fuzzy C4.5;
- transferred FCGAM;
- retrained FCGAM;

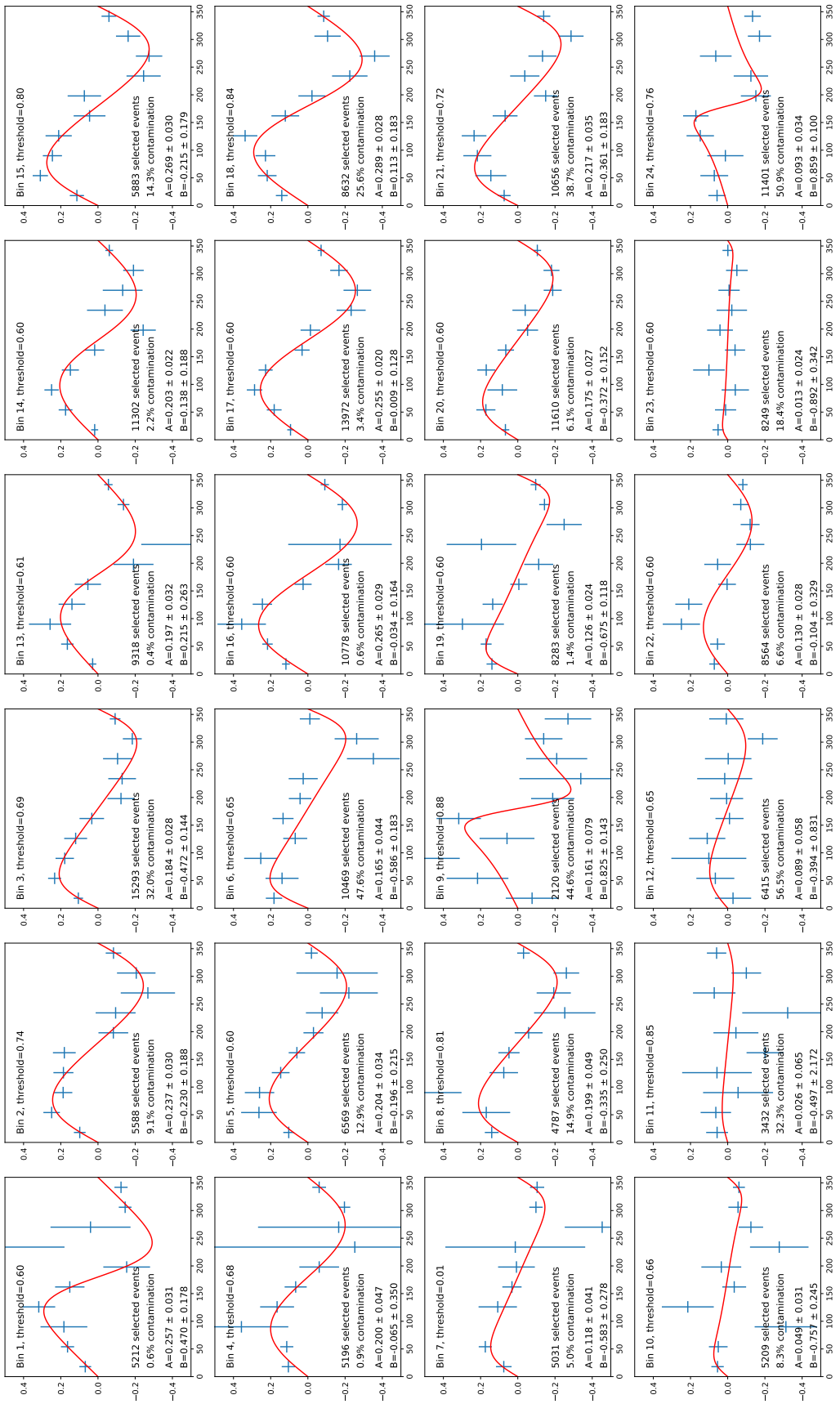


FIGURE H.1: Fuzzy C4.5 with momentum corrections.

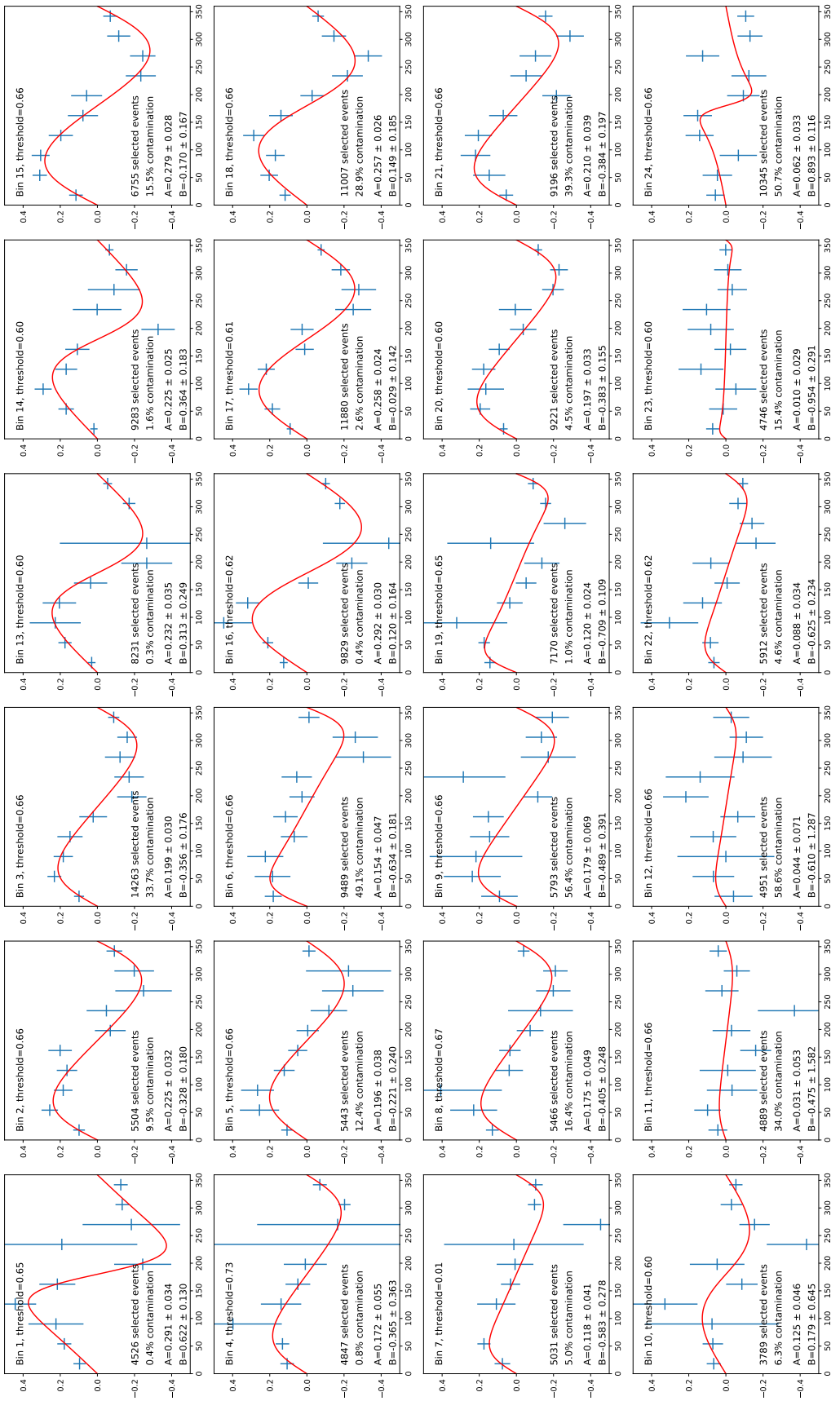


FIGURE H.2: Fuzzy C4.5 of depth 3 with momentum corrections.

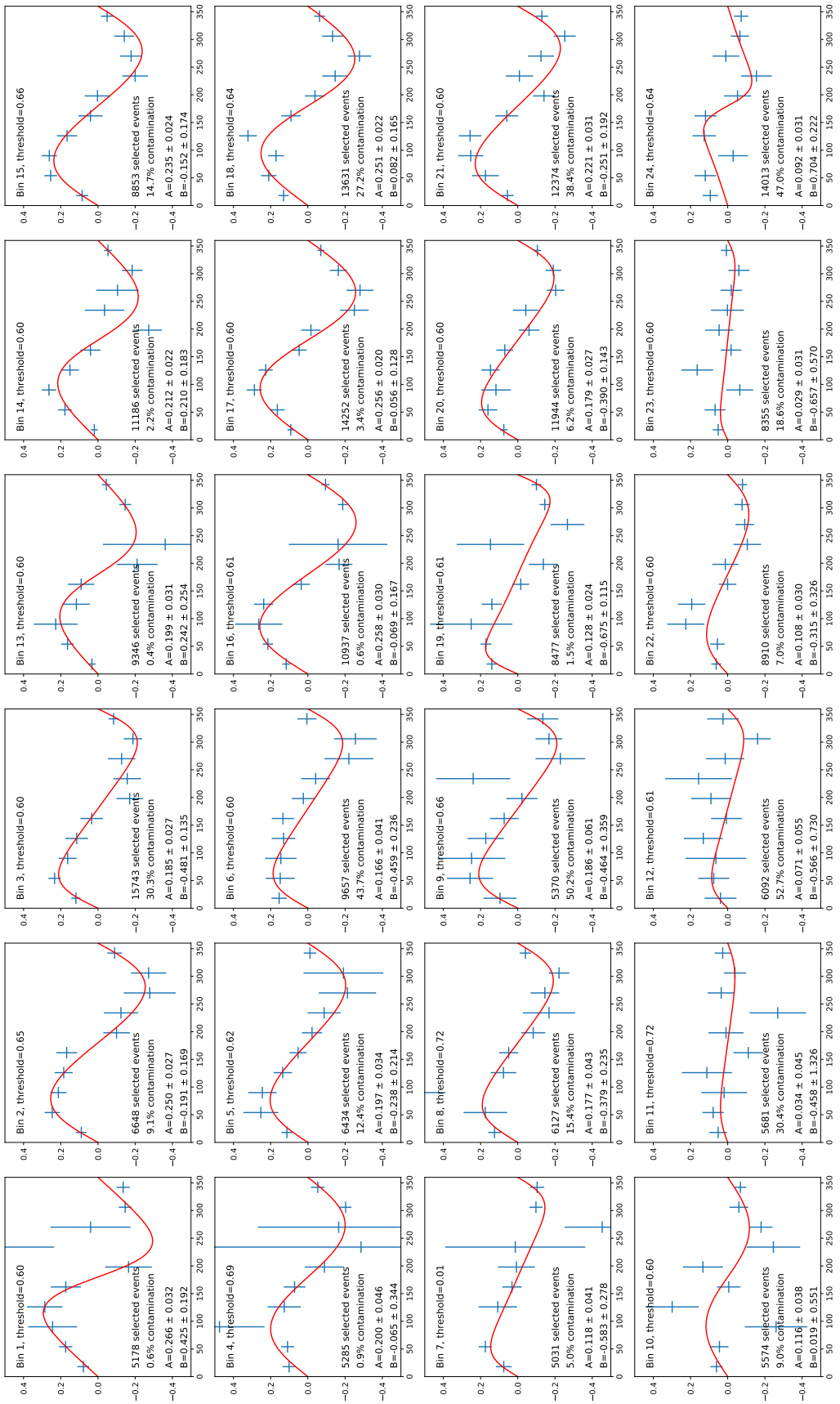


FIGURE H.3: FCGAM with momentum corrections.

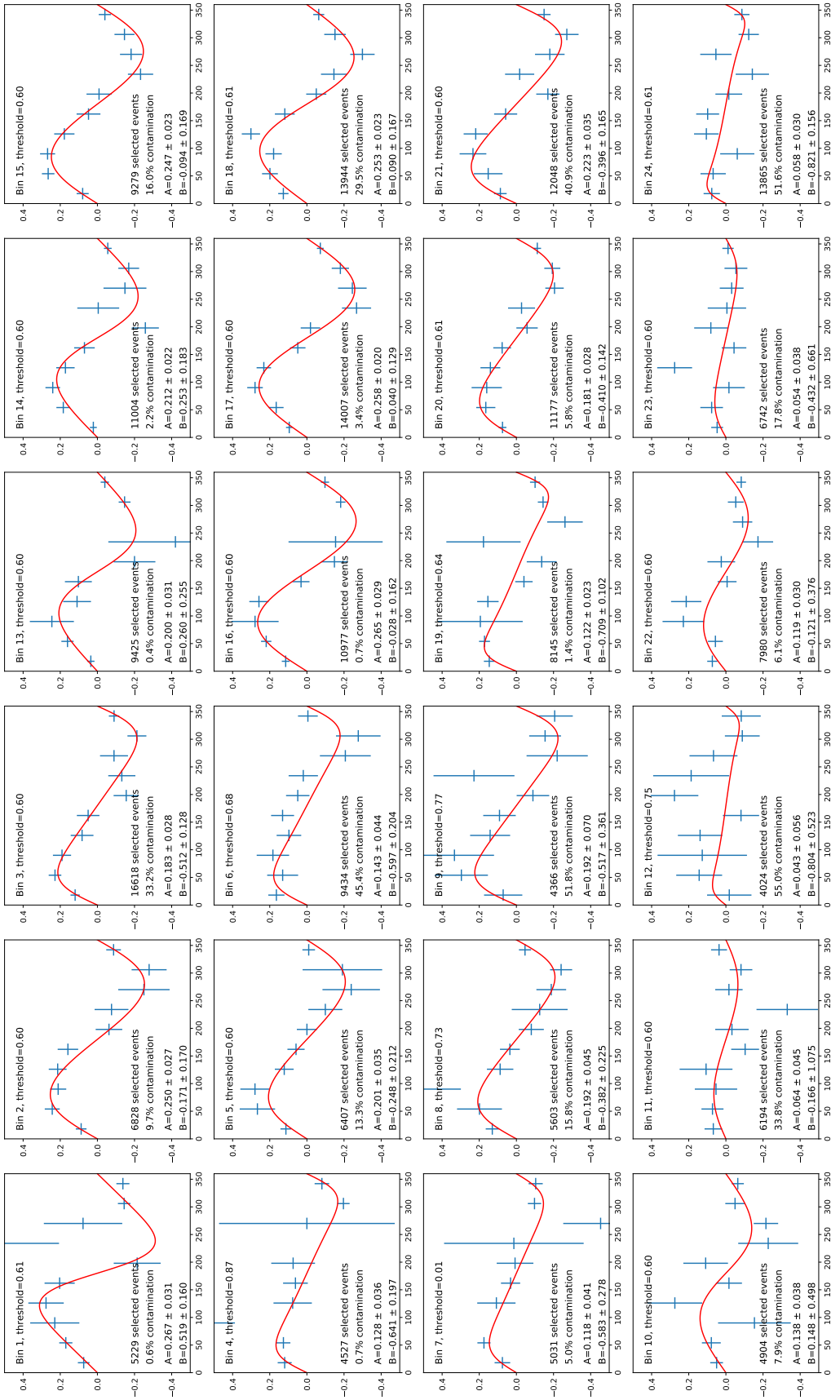


FIGURE H.4: FCGAM of 5 terms with momentum corrections.

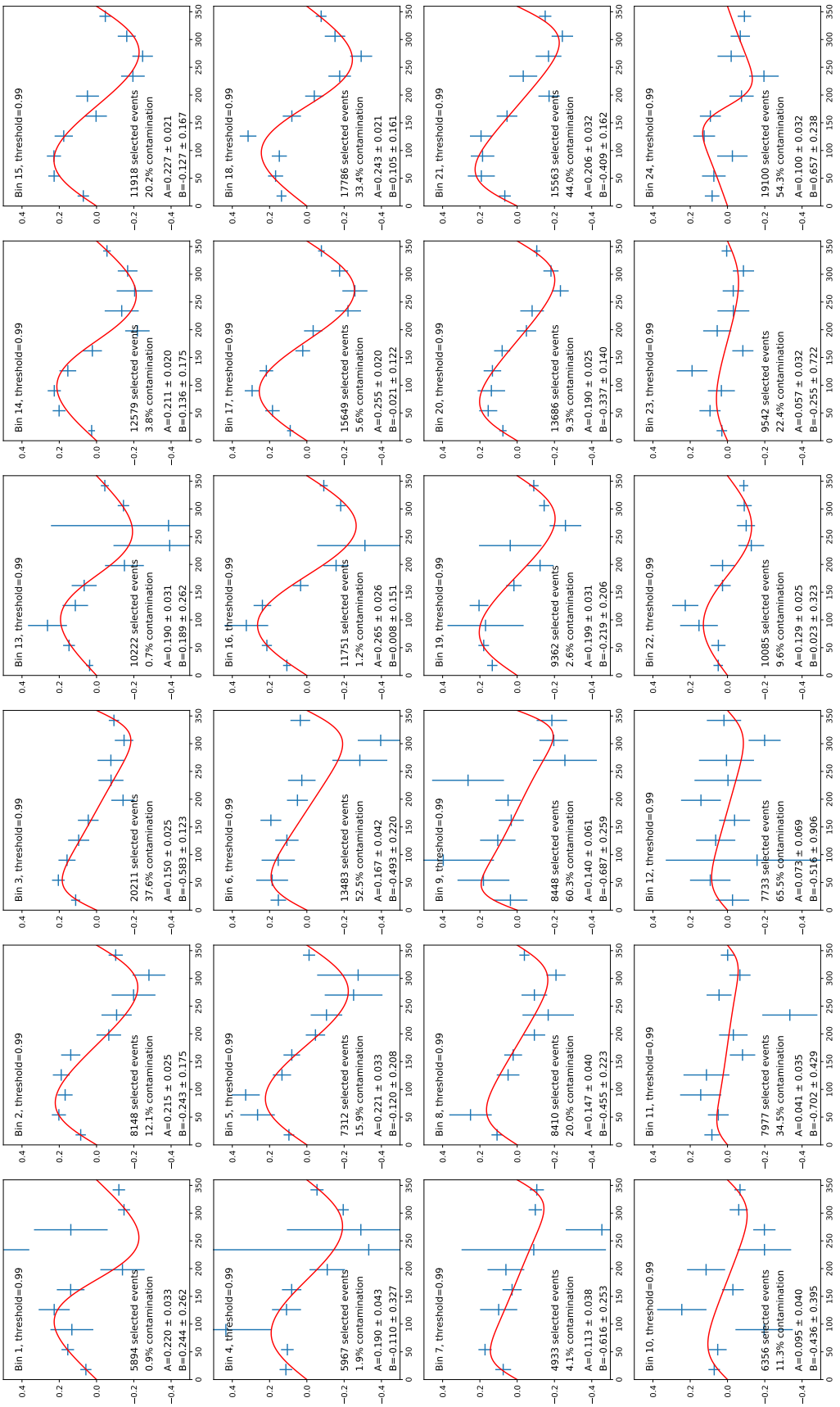


FIGURE H.5: FURIA with momentum corrections.

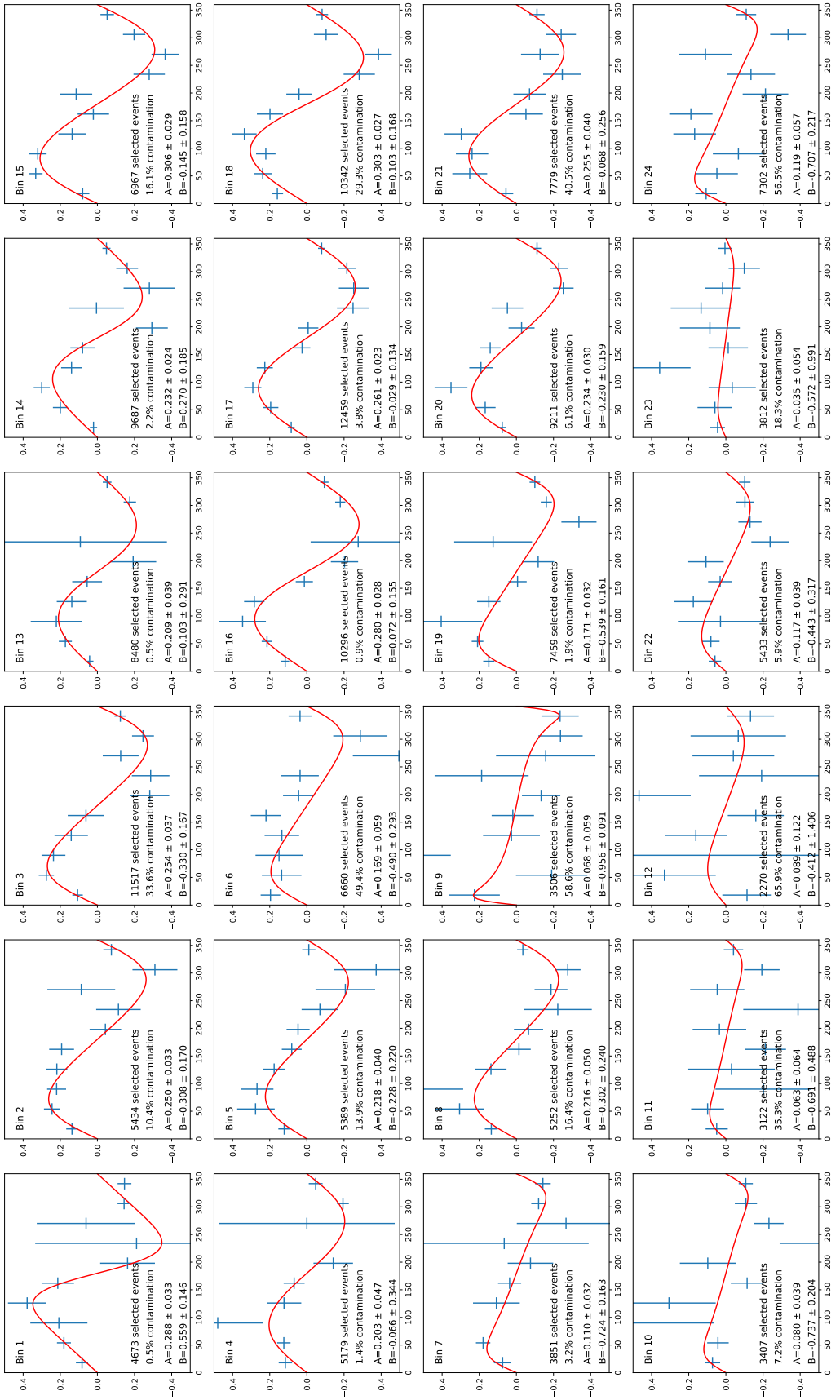


FIGURE H.6: Cuts with momentum corrections.

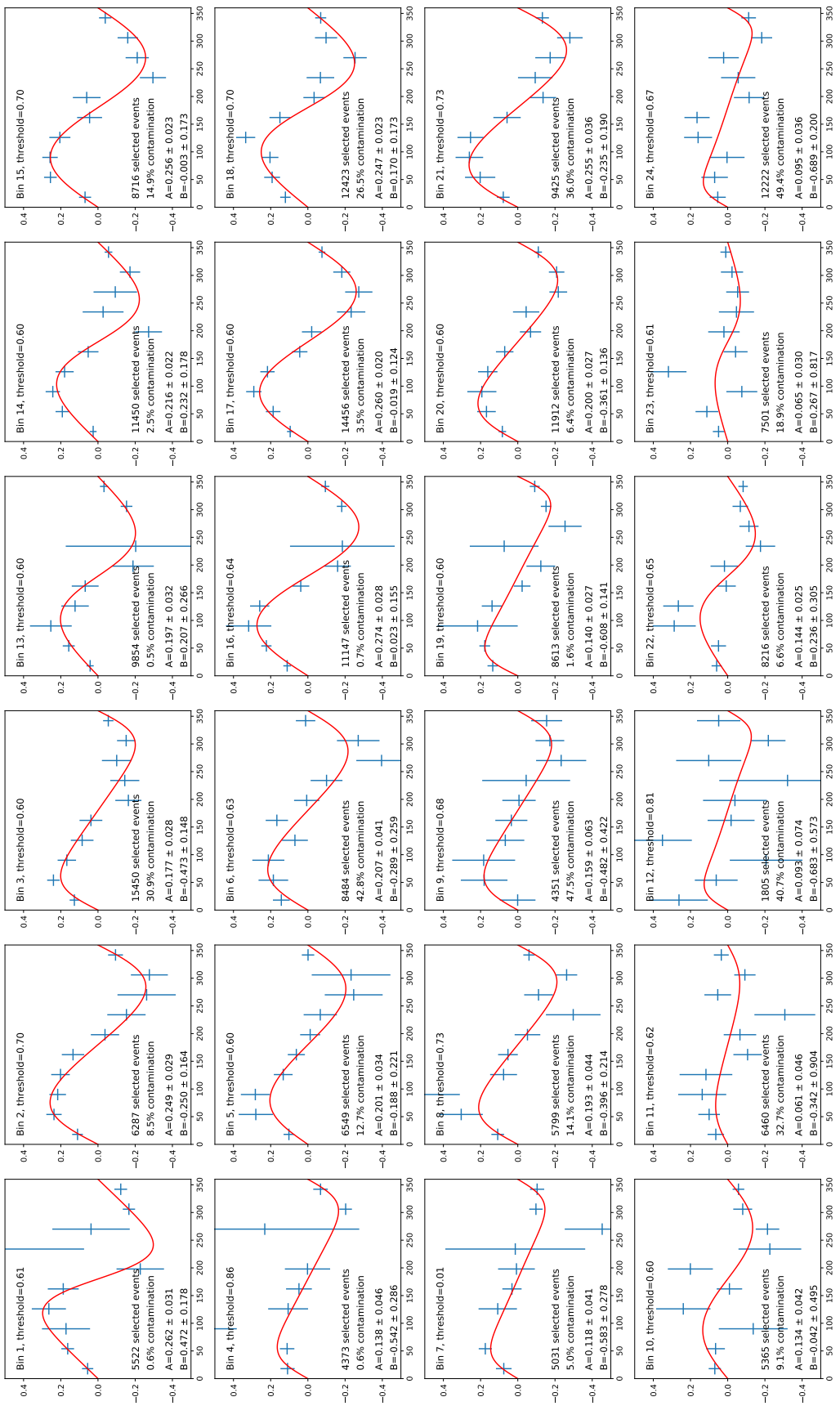


FIGURE H.7: Neural network with momentum corrections.

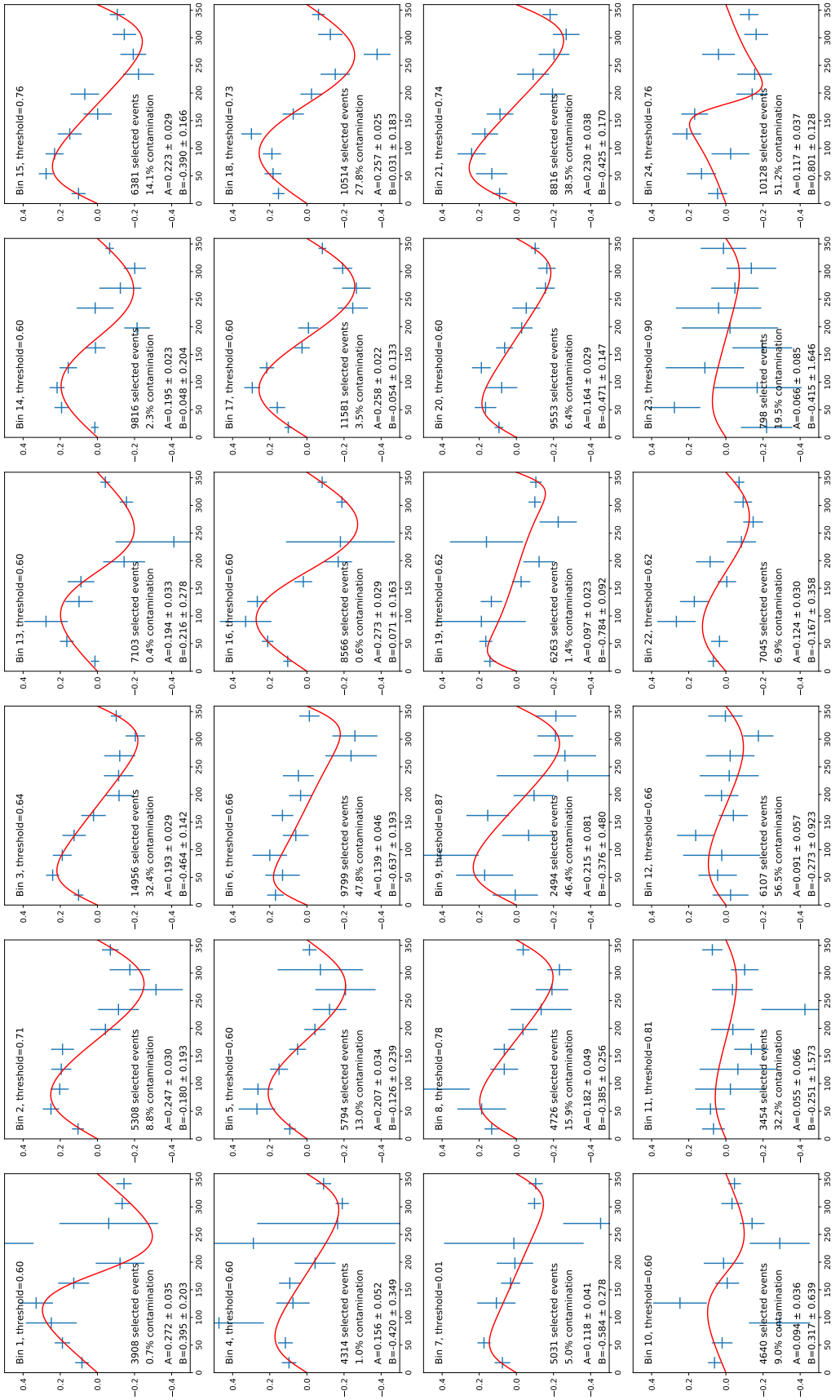


FIGURE H.8: Fuzzy C4.5 without momentum corrections.

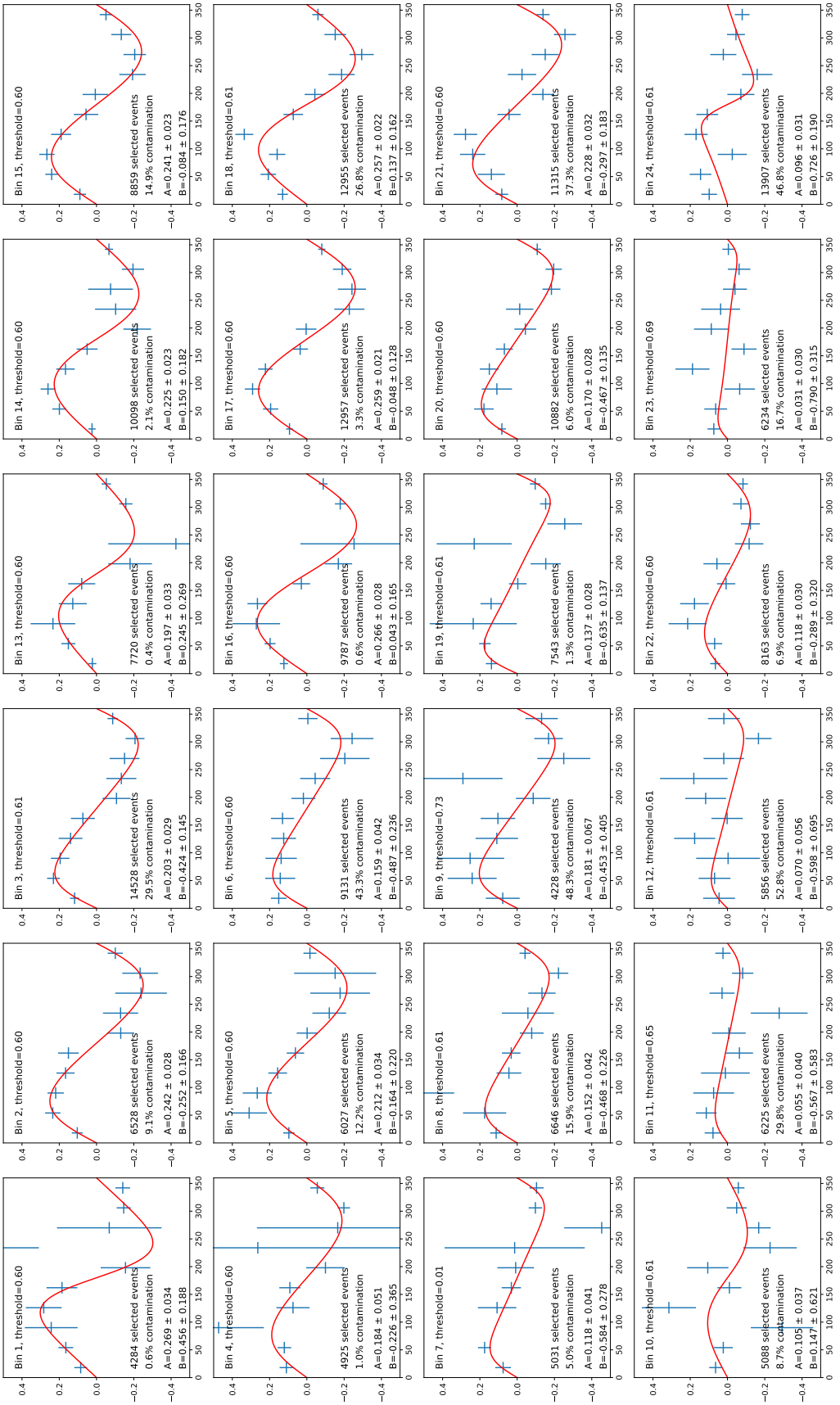


FIGURE H.9: FCGAM without momentum corrections.

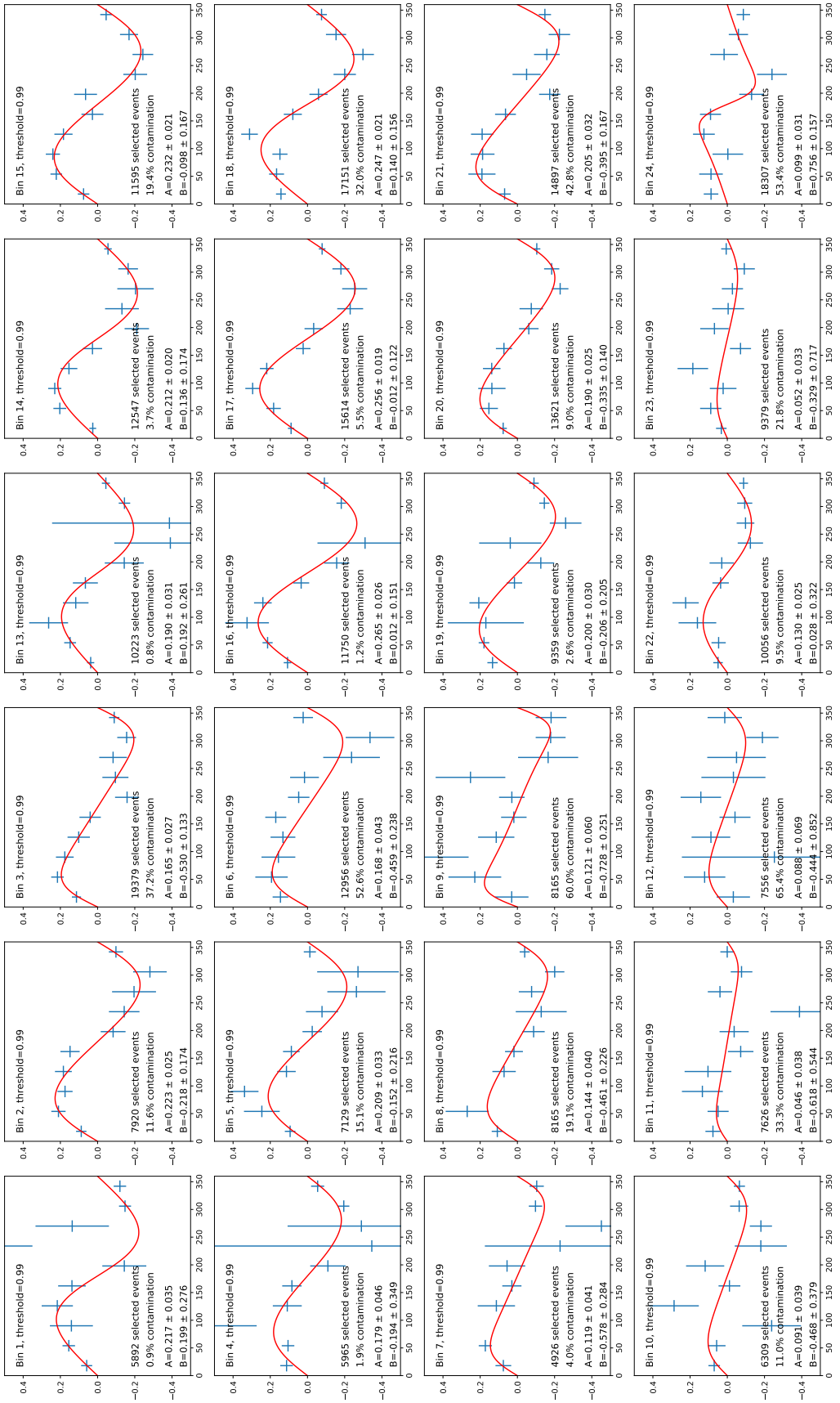


FIGURE H.10: FURIA without momentum corrections.

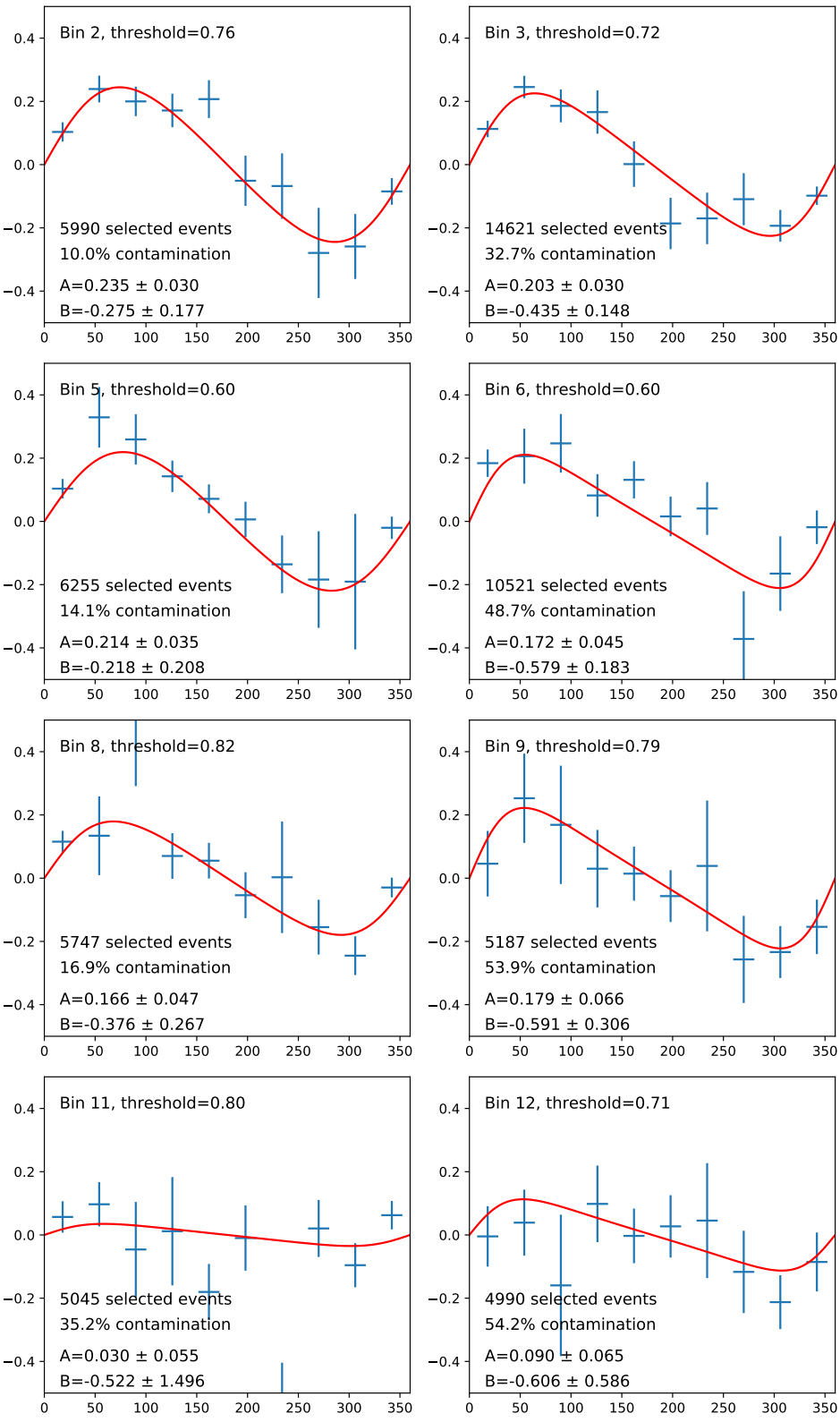


FIGURE H.11: Transferred fuzzy C4.5.

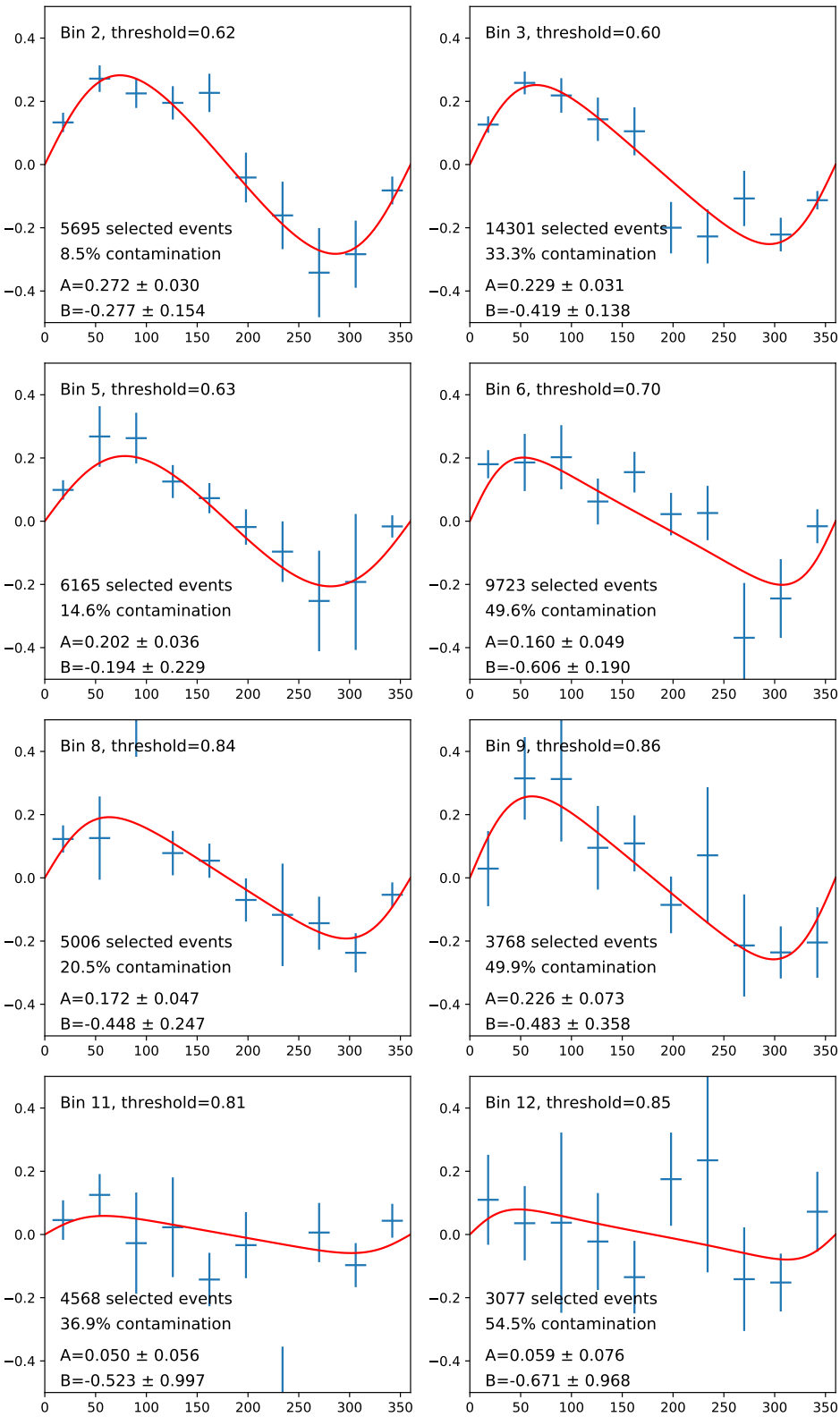


FIGURE H.12: Retrained fuzzy C4.5.

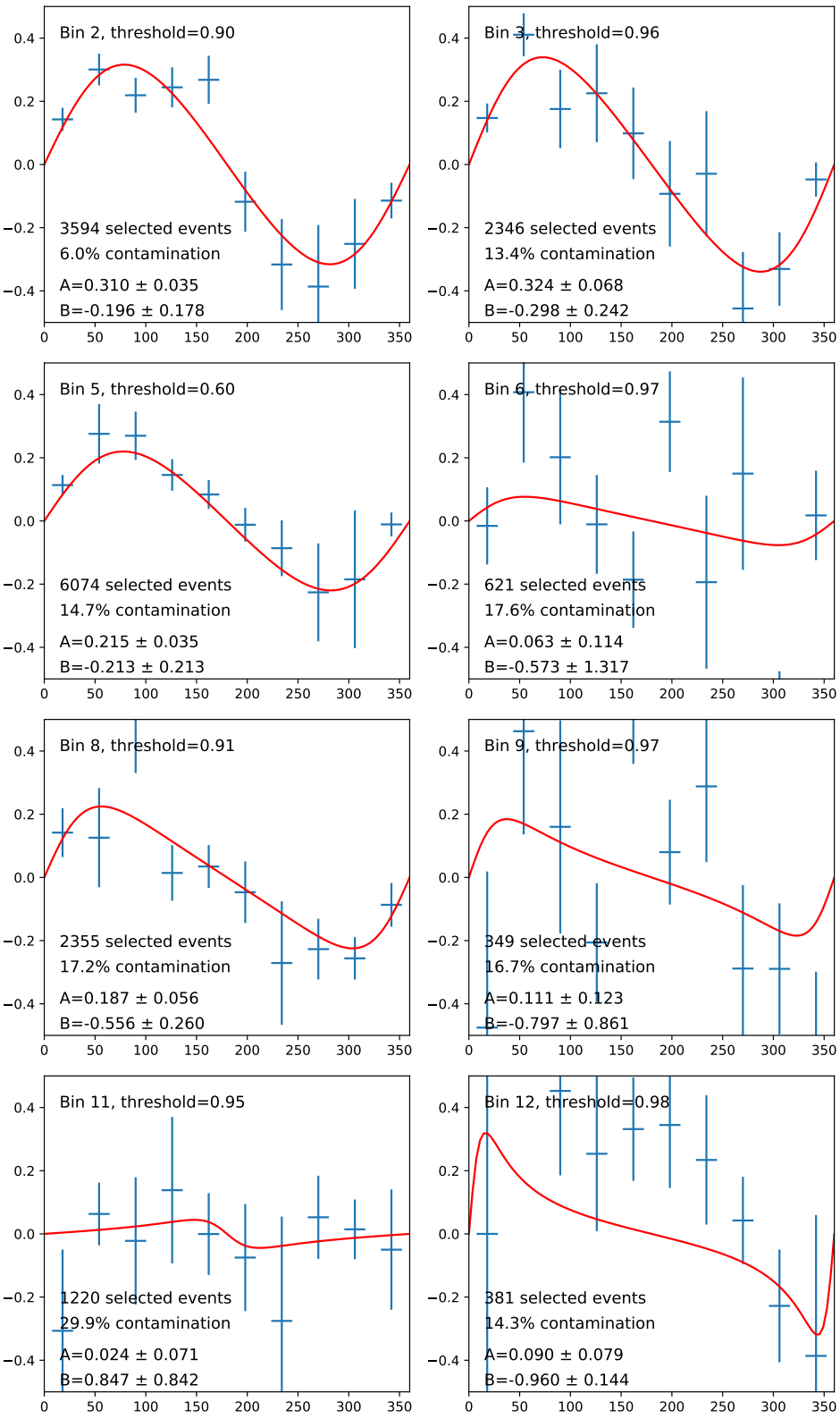


FIGURE H.13: Transferred FCGAM.

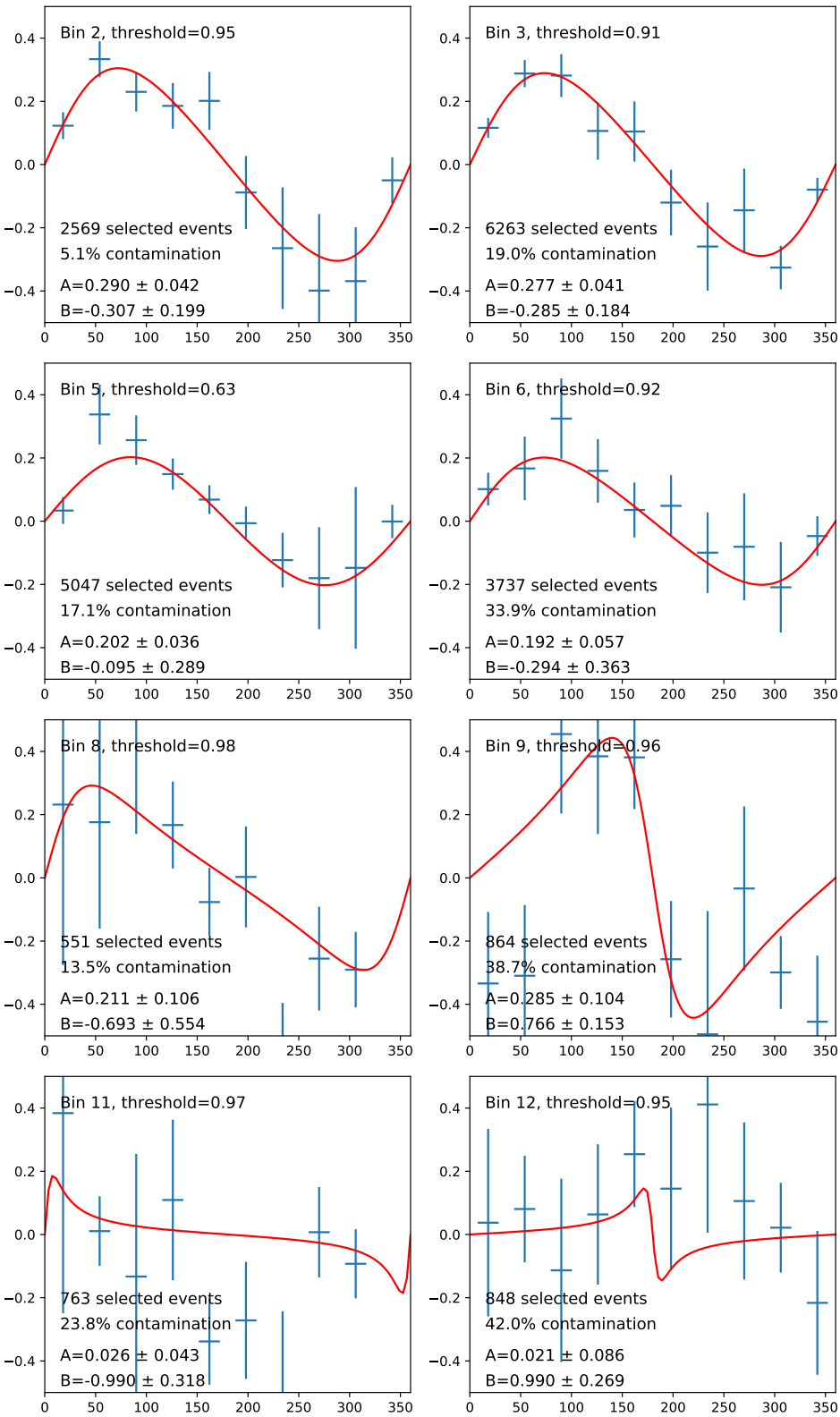


FIGURE H.14: Retrained FCGAM.

Appendix I

Résumé en français

Le reste du manuscrit ayant été rédigé en anglais, cette annexe contient un résumé en français de chacun des chapitres. Le but de cette thèse est d'analyser des données de physique expérimentale issues de l'expérience CLAS12, aux États-Unis, en utilisant des outils de machine learning interprétable. En effet, nous espérons ainsi améliorer les techniques d'analyse classiques en physique tout en restant capables de comprendre le modèle de machine learning utilisé et d'analyser les connaissances extraites des données. Cette thèse est le résultat d'une collaboration entre deux instituts du CEA : le LIST pour les problèmes liés au machine learning, et l'Irfu pour les aspects de physique expérimentale.

Partie I : Contexte et positionnement

Chapitre 1 : L'étude de la structure du proton à CLAS12

Les atomes sont les constituants élémentaires de la matière de notre univers. Ces atomes sont eux-mêmes composés d'un noyau de charge positive et d'électrons, qui sont des particules élémentaires de charge négative. Les protons et neutrons qui constituent le noyau ne sont eux pas élémentaires mais formés de quarks et gluons, liés entre eux par l'interaction forte. Le modèle standard de la physique des particules décrit l'ensemble des particules élémentaires (par exemple l'électron, les quarks et les gluons) et leurs interactions (par exemple l'interaction forte et l'interaction électromagnétique). L'interaction forte a pour particularité que ses propriétés interdisent l'existence de quarks libres à grande distance, ces derniers étant nécessairement confinés dans des *hadrons*, par exemple un proton ou un neutron. Les équations de la théorie sous-jacente de l'interaction forte deviennent très difficilement calculables à l'échelle d'un proton ou d'un neutron.

Pour pallier cette difficulté de calcul, des expériences de physique permettent d'accéder à des *fonctions de structure* décrivant l'agencement des partons (i.e. quarks et gluons) dans le proton ou dans le neutron (ici nous nous concentrons sur le cas du proton). Notamment, les *distributions de partons généralisées* (GPD) sont des fonctions de structure qui donnent accès aux corrélations entre les positions dans le plan transverse et les impulsions longitudinales des partons dans le proton. On peut accéder expérimentalement à ces GPD grâce à des processus tels que la *diffusion Compton profondément virtuelle* (DVCS), illustrée en Figure I.1. Les GPD sont liées à la polarisation de l'électron incident. Ainsi, mesurer la différence d'événements DVCS à polarisation positive ou négative (*l'asymétrie de spin du faisceau*) permet d'extraire des composantes des GPD.

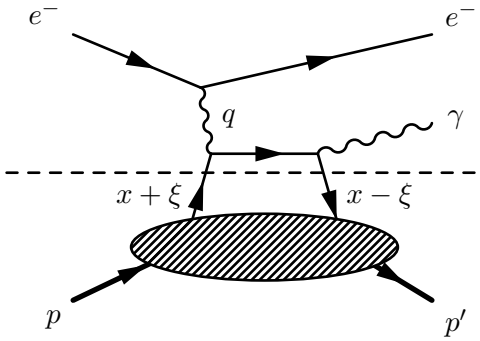


FIGURE I.1: Diagramme de Feynman de la diffusion Compton profondément virtuelle. q est l'impulsion du photon virtuel émis par l'électron, x l'impulsion longitudinale moyenne portée par le quark interagissant avec ce photon virtuel, ξ le transfert d'impulsion longitudinale.

L'expérience CLAS12 au laboratoire Jefferson a entre autres pour but de détecter des événements DVCS et d'en mesurer des observables physiques liées aux GPD. Un faisceau d'électrons porté à 10,6 GeV est envoyé sur une cible fixe de protons. Le détecteur CLAS12 a pour but de détecter les particules issues des collisions électron-proton.

Une méthode de sélection d'événements doit être conçue afin d'isoler les événements DVCS des autres processus qui constituent un bruit de fond. Classiquement, les physiciens utilisent des variables qui vérifient la conservation de l'impulsion. Par exemple, la masse manquante au carré $ep \to ep\gamma X$ devrait être proche de 0 pour un événement DVCS car aucune autre particule n'a été produite :

$$M^2_{ep\to ep\gamma X} = \|p_{e,in} + p_{p,in} - p_{e,out} - p_{p,out} - p_{\gamma,out}\|^2 \tag{I.1}$$

avec $p_{e,in}$, $p_{p,in}$, $p_{e,out}$, $p_{p,out}$, $p_{\gamma,out}$ l'impulsion de respectivement l'électron, le proton d'entrée, et l'électron, proton et photon de sortie. Grâce à cette sélection d'événements, l'asymétrie DVCS peut être calculée. L'objectif de cette thèse est d'améliorer l'étape de sélection d'événements avec des outils de machine learning interprétable. En assurant l'interprétabilité des modèles utilisés, la validation de l'analyse par les pairs pour publication est facilitée, et nous espérons idéalement pouvoir analyser les connaissances extraites des données par le modèle pour améliorer les techniques existantes de sélection d'événements.

Chapitre 2 : Machine learning interprétable

L'interprétabilité en machine learning est un sujet d'intérêt grandissant dans la littérature ces dix dernières années. L'interprétabilité en machine learning se définit comme « la capacité à expliquer ou à présenter en termes compréhensibles à un humain ». En particulier, l'évaluation de l'interprétabilité d'un modèle est complexe. S'il existe des critères objectifs d'interprétabilité tels que la complexité du modèle, une étude plus complète sollicitant des humains doit être conduite pour parfaire l'évaluation, notamment dans le cadre d'applications précises. Parmi les techniques d'interprétation de modèles de machine learning, on trouve d'une part les techniques d'explication a posteriori d'un modèle opaque, et d'autre part les modèles intrinsèquement transparents, qui ne nécessitent pas de plus amples explications.

Dans ce chapitre, nous nous concentrons sur trois familles principales de modèles transparents :

- Les arbres de décisions sont des modèles utilisés à la fois pour la classification et la régression. Leurs feuilles contiennent l'information pour calculer la prédiction, alors que leurs nœuds comprennent les conditions sur les attributs pour diriger vers un des nœuds fils. Les arbres de décision peuvent être catégoriels (ID3 notamment) ou numériques (par exemple CART et C4.5) selon la nature des données qu'ils peuvent prendre en compte et leur façon de les diviser ;
- Les bases de règles contiennent une liste ordonnée ou non de règles sous la forme SI *conditions* ALORS *conclusion*. Elles sont majoritairement utilisées pour la classification ;
- Les modèles additifs généralisés (GAM) sont des modèles statistiques dérivés des modèles linéaires généralisés et des modèles additifs. Ils s'écrivent comme la somme de fonctions d'une variable :

$$g(\widetilde{y}) = \beta_0 + \sum_j f_j(x_j) \tag{I.2}$$

avec g une fonction de lien (une sigmoïde inverse pour la classification), $\widetilde{y}$ la variable prédite, β_0 une constante à inférer, f_j les fonctions à inférer, x_j les attributs. Un GAM est illustré sur la Figure I.2.

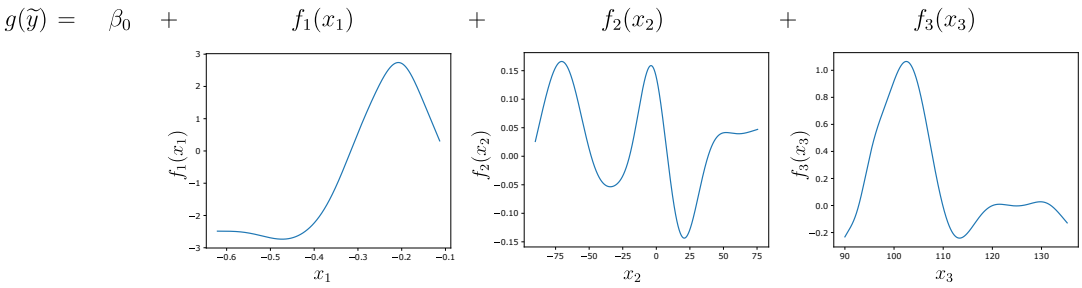


FIGURE I.2: Un GAM comprenant trois termes.

Les deux premières familles de modèles admettent des variantes floues : au lieu de considérer les tests sur les variables de manière booléenne, on attribue à chaque instance un degré d'appartenance à une prémisse, une règle ou à un nœud ou une branche d'un arbre.

L'inconvénient des modèles transparents par rapport à d'autres modèles plus opaque est souvent leur moindre performance de classification. C'est pourquoi cette thèse cherche notamment à combler cette lacune.

Chapitre 3 : Simulation CLAS12 et premiers résultats avec des modèles transparents

L'entraînement de modèles de machine learning nécessite des données labellisées. Nous utilisons le logiciel de simulation de CLAS12, dans lequel nous injectons des événements DVCS et des événements de production de π^0, le principal processus constituant le bruit de fond. En utilisant FURIA, un algorithme d'induction d'une base de règles

floue, sur deux jeux de données générés différemment, nous découvrons qu'il est important de générer des données non biaisées par rapport aux sections efficaces. Dans le cas contraire, le modèle apprend à séparer les processus en utilisant leurs distributions dans l'espace de phase au lieu de raisonner sur des lois physiques de conservation.

Dans la suite de la thèse, nous considérerons plusieurs modèles en particulier : un arbre C4.5 et deux variantes floues, deux ensembles d'arbres (AdaBoost et Gradient-Boosting), une base de règles floue induite par FURIA, et un GAM. Les résultats de premiers tests sur les données simulées indiquent que les arbres C4.5 flous et le GAM seraient a priori les meilleurs modèles en termes de performances de classification.

Partie II : Machine learning interprétable grâce à la construction de variables

Chapitre 4 : État de l'art des méthodes de construction de variables

La performance des algorithmes de machine learning dépend largement des attributs fournis en entrée. C'est en particulier le cas des algorithmes les moins complexes, ce qui concerne donc souvent les algorithmes d'induction des modèles transparents auxquels cette thèse s'intéresse. Pour compenser cette limitation, la construction automatique de variables permet de concevoir une nouvelle représentation des données, plus pertinente vis à vis de la tâche de classification ou de régression considérée. Nous nous concentrons ici sur la construction explicite de variables à l'aide d'opérateurs notamment numériques (+, −, log, etc.).

Les algorithmes de construction de variables se distinguent notamment par leur méthode d'évaluation des variables candidates:

- les méthodes *filtre* calculent un score pour chaque variable candidate indépendamment d'un quelconque algorithme d'apprentissage, par exemple avec le gain d'information, ou la corrélation avec la variable cible ;
- les méthodes dites *« wrapper »* utilisent le score de prédiction d'un algorithme d'apprentissage pour évaluer les variables candidates ;
- les méthodes *intégrées* combinent la phase de sélection et construction de variables avec l'apprentissage en lui-même : ces méthodes sont très spécifiques à l'algorithme d'apprentissage considéré.

Les techniques de construction de variables se divisent en deux grandes catégories. Historiquement, les premiers travaux sur la construction de variables utilisent des méthodes à base d'arbres pour explorer l'espace de recherche. Le principe est de générer itérativement de nouvelles variables en combinant celles de l'étape précédente avec de nouveaux opérateurs. Une autre classe de méthodes utilise les algorithmes évolutionnaires. Le principe est de faire évoluer une population d'individus qui encodent les variables candidates dans leurs gènes. Deux familles d'algorithmes évolutionnaires sont utilisées pour la construction automatique de variables : l'intelligence distribuée et la programmation génétique, cette dernière étant de loin la plus représentée dans la littérature. En programmation génétique, les individus sont sous forme d'arbre où les nœuds sont des opérateurs et les feuilles les attributs initiaux.

Chapitre 5 : Construction de variables interprétables en tant que méthode en amont

Dans ce chapitre, nous proposons une méthode pour contraindre l'algorithme classique de programmation génétique de sorte à obtenir des variables construites qui respectent les unités physiques des attributs initiaux et qui ont un sens physique.

D'une part, nous utilisons une *grammaire* pour contraindre l'évolution de l'algorithme de programmation génétique. Ici, une grammaire non contextuelle est définie pour des applications en physique des hautes énergies, qui liste les opérations autorisées (par exemple additionner deux énergies est autorisé, au contraire de la soustraction d'un angle et d'une impulsion). D'autre part, une matrice de transition définit la distribution de probabilités du choix d'un opérateur à la suite d'un autre. Cette matrice permet de guider la recherche de variables vers des variables qui ressemblent à des variables usuelles utilisées par les physiciens. Par exemple, une racine carrée est souvent suivie d'une somme de carrés (norme d'une impulsion en physique pour vérifier les lois de conservation).

Lors d'expérimentations sur trois jeux de données de physique des hautes énergies, le gain d'information, le score de classification de CART et celui de FURIA sont successivement utilisés comme fonctions de fitness pour l'évolution de la programmation génétique. Dans tous les cas, une augmentation significative du score de classification est obtenue sur les trois jeux de données considérés, jusqu'à 60% en utilisant FURIA. La programmation génétique contrainte (avec la grammaire et la matrice de transition) permet également d'avoir un meilleur score par rapport à l'algorithme de base de programmation génétique. Enfin, nous avons discuté de l'interprétabilité des variables produites avec des arguments physiques : les caractéristiques de ces variables ont pu être reliées la plupart du temps à une explication physique, qu'elle soit liée aux lois de conservation ou à des considérations géométriques. Par exemple, une des variables les plus fréquemment construites pour CLAS12 est la somme des composantes en z des impulsions des trois particules de sortie du DVCS : $p_z^e + p_z^p + p_z^{\gamma_1}$, ce qui correspond à la conservation des impulsions sur l'axe z.

Chapitre 6 : Construction de variables interprétables intégrée

Dans le chapitre précédent, la construction de variables était réalisée en amont de l'induction du modèle de classification. En utilisant des fonctions de fitness telles que le score de CART ou FURIA, le temps de calcul devient rapidement très important. Dans ce chapitre, nous proposons d'intégrer l'algorithme de construction de variables contraint avec la grammaire et la matrice de transition dans l'induction des modèles.

Dans un premier temps, nous intégrons la construction de variables dans les algorithmes d'induction d'arbres de décision et de bases de règles. L'idée est d'utiliser le critère de sélection de l'attribut et de son seuil à chaque nœud de l'arbre ou à chaque prémisse de règle comme fonction de fitness pour l'algorithme de programmation génétique. Pour C4.5, il s'agit du gain d'information. Nous pouvons soit construire une variable à chaque nœud de l'arbre ou chaque prémisse de chaque règle, soit nous limiter aux premiers N_{max} nœuds en partant de la racine ou aux premières N_{max} prémisses de chaque règle constituant la base. Des expérimentations ont été réalisées sur quatre jeux de données de physique des hautes énergies, intégrant la construction de variables

en particulier dans C4.5 et FURIA, mais également dans les versions floues de C4.5, dans CART, et dans AdaBoost et GradientBoosting. Dans tous les cas, nous observons une amélioration significative du score de classification par rapport au modèle de base sans construction de variables. Comme illustré en Figure I.3 pour C4.5 et FURIA, l'augmentation de ce score en fonction du nombre de variables construites atteint un plateau après un certain seuil sur N_{max}, ce qui semble démontrer qu'on peut améliorer la performance de classification de tels modèles tout en limitant leur complexité, ce qui est un argument en faveur de leur facilité d'interprétation.

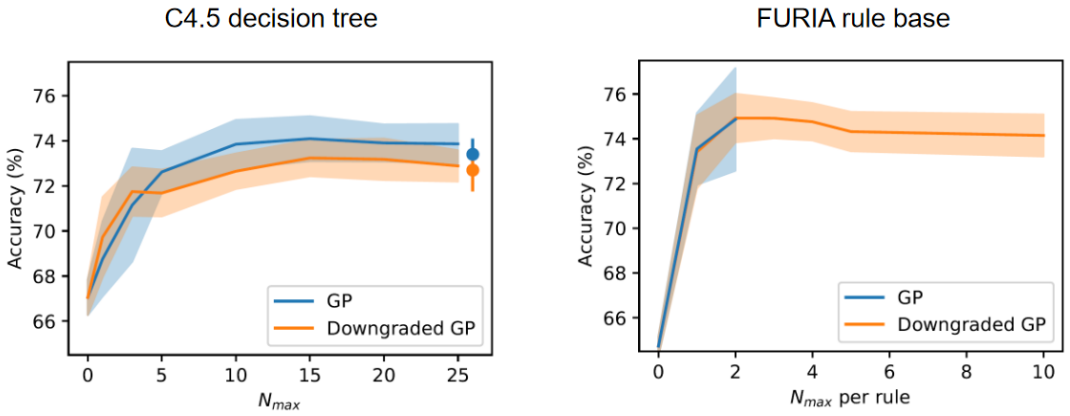


FIGURE I.3: Évolution de la précision de classification obtenue par des modèles C4.5 et FURIA sur les données de simulation CLAS12 en fonction du nombre de variables construites pendant leur induction.

Dans un second temps, nous avons intégré la construction de variables dans les GAM. Contrairement aux arbres de décision et bases de règles, il n'existe pas de critère de sélection des attributs les plus pertinents dans les GAM. Ainsi, nous proposons une méthode d'induction des GAM basée sur le principe de gradient boosting : les termes sont ajoutés un à un de sorte à minimiser les résidus à chaque étape. Pour chaque terme, une variable est construite avec l'algorithme de programmation génétique avec pour fonction de fitness le score de classification d'un arbre de décision réduit à quatre feuilles (profondeur de 2). D'autre part, nous introduisons dans ce chapitre une contrainte additionnelle sur les fonctions des termes du GAM de sorte à renforcer l'interprétabilité et le pouvoir de généralisation. En effet, les variables discriminantes en physique ont souvent une relation *bitonique* par rapport à la variable cible : une masse manquante présente un pic à zéro ou à la masse de la particule manquante. Ainsi, nous proposons une définition et une méthode d'évaluation du degré de bitonicité d'une variable. Ensuite, nous proposons une méthode pour contraindre la construction de variables et/ou l'induction des fonctions des termes du GAM à produire des résultats bitoniques. L'application des contraintes de bitonicité permet d'améliorer l'interprétabilité tout en n'impactant pas significativement le score de classification.

Partie III : De la simulation à l'analyse des données réelles de CLAS12

Chapitre 7 : Transfert des modèles aux données réelles

Ce chapitre a pour but de traiter les différences de distributions existantes entre les données simulées et les données réelles de CLAS12. Pour fonctionner de manière optimale, les modèles de machine learning entraînés sur les données simulées doivent être adaptés aux données réelles. À CLAS12, plusieurs sources d'erreurs sur les distributions existent, incluant les incertitudes sur les sections efficaces, sur les résolutions des détecteurs, et sur la géométrie globale. Ne disposant pas d'un échantillon labellisé représentatif des données réelles, nous nous concentrons sur l'adaptation de domaine plutôt que sur l'apprentissage par transfert.

Il existe de nombreuses approches d'adaptation de domaine dans la littérature. On s'intéresse en particulier aux méthodes qui cherchent à établir une transformation entre les domaines source et cible. Parmi ces méthodes, le transport optimal et l'apprentissage profond sont deux approches plébiscitées.

La flexibilité que permet l'apprentissage profond nous pousse à adopter cette approche pour l'adaptation de domaine. Notamment, les modèles génératifs adversariaux sont de bons candidats. Une transformation est apprise pour chaque particule (électron, proton, photons), et un discriminateur permet de vérifier si l'ensemble est compatible avec les distributions réelles.

On utilise les événements de production π^0 exclusifs, faciles à simuler d'une part et à isoler dans les données d'autre part, comme références pour apprendre la transformation. Les expériences sur des données simulées indépendamment des sections efficaces permettent de valider la pertinence de l'approche proposée. Sur des données avec sections efficaces, nous remarquons que la généralisation aux événements DVCS notamment est perfectible. En effet, les événements de production π^0 utilisés ne couvrent qu'une partie limitée de l'espace de phases. Une méthodologie est proposée dans les perspectives du chapitre pour surmonter cette limitation. Les modèles de machine learning transparents peuvent ensuite être transférés aux données simulées transformées.

Chapitre 8 : Évaluation de l'interprétabilité par des physiciens expérimentaux

Une évaluation complète de l'interprétabilité d'un modèle nécessite de recueillir l'avis des utilisateurs finaux, ici des physiciens expérimentaux. Nous avons conçu un sondage à destination des physiciens afin d'évaluer différents aspects des modèles proposés : la compréhension des variables construites automatiquement par la programmation génétique, l'interprétabilité des modèles eux-mêmes (GAM, FURIA), et une comparaison globale de plusieurs approches pour la sélection d'événements. 6 bêta-testeurs puis 25 physiciens des collaborations CLAS12, Hall A et COMPASS ont répondu au sondage.

Les résultats concernant les variables automatiquement construites montrent que les contraintes ajoutées à l'algorithme de programmation génétique permettent de significativement augmenter l'interprétabilité des variables construites. Nous n'observons

pas de différence significative entre les variables construites en amont ou intégrées à l'induction des modèles, ni entre les variables construites automatiquement et les variables usuelles utilisées par les physiciens pour la sélection d'événements.

L'interprétabilité de trois modèles de machine learning a été évaluée : un GAM, une base induite par l'algorithme FURIA, et un réseau de neurones. La compréhension du fonctionnement des modèles par les répondants a d'abord été mesurée : le GAM et la base induite par FURIA ont été globalement bien compris. Un score de transparence a été demandé en complément : le GAM et FURIA obtiennent un score semblable, supérieur au score obtenu par le réseau de neurones.

Enfin, les répondants ont été sollicités pour classer quatre modèles entre eux pour la sélection d'événements : un GAM, une base induite par FURIA, un réseau de neurones et un ensemble de coupures utilisées par les physiciens. Le GAM obtient le meilleur rang en moyenne, et est souvent classé deuxième derrière soit le réseau de neurones soit l'ensemble de coupures. Les GAM semblent donc constituer un bon compromis entre interprétabilité et performance, satisfaisant la majorité des physiciens.

Chapitre 9 : Analyse des données DVCS de CLAS12

Ce dernier chapitre détaille l'application des modèles élaborés jusqu'ici aux données réelles de CLAS12. Des corrections d'impulsions sont appliquées pour corriger les différences majeures de distributions entre les données simulées et réelles. Une poignée de modèles est sélectionnée pour l'analyse : un GAM, un arbre C4.5 flou, une base induite par FURIA. Nous nous assurons de la bonne suppression notamment des événements de production η dans les données CLAS12 présélectionnées.

Une méthode de soustraction des événements de production π^0 permet d'estimer la proportion de ces événements dans les données sélectionnées. La Figure I.4 illustre la distribution des valeurs de sortie du GAM pour l'ensemble des données présélectionnées et pour la contamination π^0.

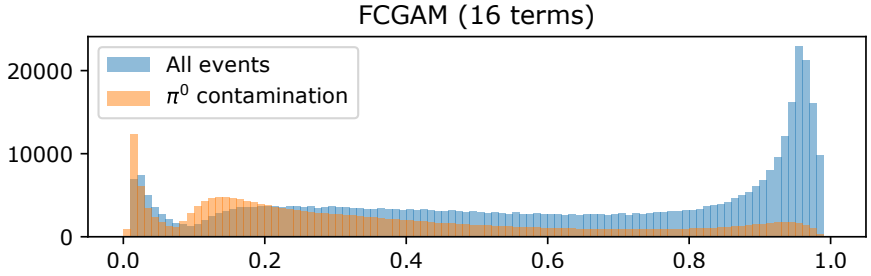


FIGURE I.4: Distribution des valeurs prédites par le GAM pour les données présélectionnées, et contamination π^0 estimée.

Le but est ensuite de choisir le seuil de coupure sur la sortie du GAM. Ce seuil est choisi de façon à minimiser l'erreur statistique sur l'asymétrie, l'observable physique qui nous intéresse. Nous obtenons par exemple la courbe d'asymétrie en fonction de la variable cinématique ϕ en Figure I.5 pour les bins cinématiques 2, 3 et 17.

Enfin, nous avons comparé cinq approches pour la sélection d'événements à CLAS12 : le GAM, la base induite par FURIA, l'arbre C4.5 flou, un réseau de neurones et un ensemble de coupures, en termes d'erreur statistique sur l'asymétrie. Le GAM surpasse le réseau de neurones dans la majorité des bins cinématiques. Cela conclut

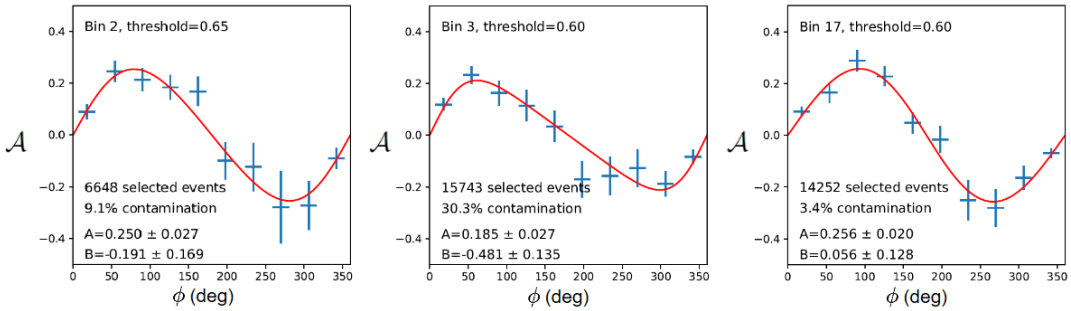


FIGURE I.5: Asymétries dans les bins 2, 3 et 17 en fonction du paramètre cinématique ϕ, calculées en utilisant le GAM.

positivement notre analyse puisque l'on a élevé les performances du GAM, grâce à la construction de variables et à la bitonicité, au niveau du réseau de neurones considéré. Dans le même temps nous avons conservé voire amélioré l'interprétabilité du GAM, qualité dont ne dispose pas le réseau de neurones. De plus, le GAM permet de sélectionner 24% d'événements DVCS supplémentaires par rapport à l'ensemble de coupures classiquement utilisées par les physiciens, ce qui confirme la pertinence des modèles de machine learning pour améliorer l'analyse.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Particules, hadrons, énergie et noyau : instrumentation, imagerie, cosmos et simulation (PHENIICS)

Titre: Machine learning interprétable pour l'analyse de données à CLAS12

Mots clés: Apprentissage automatique, Analyse de données, Physique hadronique

Résumé: L'intelligence artificielle rencontre un succès indéniable dans de nombreuses applications, surtout depuis l'essor de l'apprentissage profond. Cependant, certaines de ces applications nécessitent une étude et une validation du raisonnement du modèle induit. C'est le cas en physique expérimentale : les performances des modèles sur les données réelles doivent être connues et maîtrisées, et leur raisonnement expliqué afin de permettre une validation par les pairs. Dans le cas particulier de l'expérience CLAS12 au Jefferson Laboratory, un faisceau d'électrons est envoyé sur une cible de protons afin d'en sonder la structure interne. Pour pouvoir accéder à certaines fonctions de structure du proton, un sous-ensemble des données récoltées doit être isolé correspondant à une réaction exclusive : la diffusion Compton profondément virtuelle. C'est sur la sélection de ces événements que porte cette thèse. Pour améliorer les techniques classiques d'analyse en physique, une approche utilisant des modèles de machine learning intrinsèquement interprétables, dits également transparents, est proposée. De cette façon, le fonctionnement du modèle peut être plus facilement compris et les erreurs de sélection maîtrisées et minimisées.

Title: Interpretable machine learning for CLAS12 data analysis

Keywords: Machine learning, Data analysis, Hadronic physics

Abstract: Artificial intelligence is used massively in numerous applications, especially since the rise of deep learning techniques. However, some of these applications require a careful study and validation of the inducted model functioning. Considering experimental physics, the performances of the models on real data must be known and controlled, and their functioning explained to enable a validation via peer review. In the particular case of the CLAS12 experiment at Jefferson Laboratory, an electron beam is sent onto a proton target to probe its inner structure. To access certain structure functions of the proton, a subset of the collected data must be selected corresponding to an exclusive interaction: deeply virtual Compton scattering. This thesis focuses on this event selection. To improve the classical physics analysis, an approach exploiting intrinsically interpretable machine learning models, also called transparent models, is proposed. In this way, the functioning of the model is understood more easily and the selection errors are minimized and controlled.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France