INTERPRETABLE MACHINE LEARNING TECHNIQUES FOR CLAS12 DATA ANALYSIS

Artificial intelligence and physics | Noëlie Cherrier

OUTLINE

1. Introduction to the context and problem

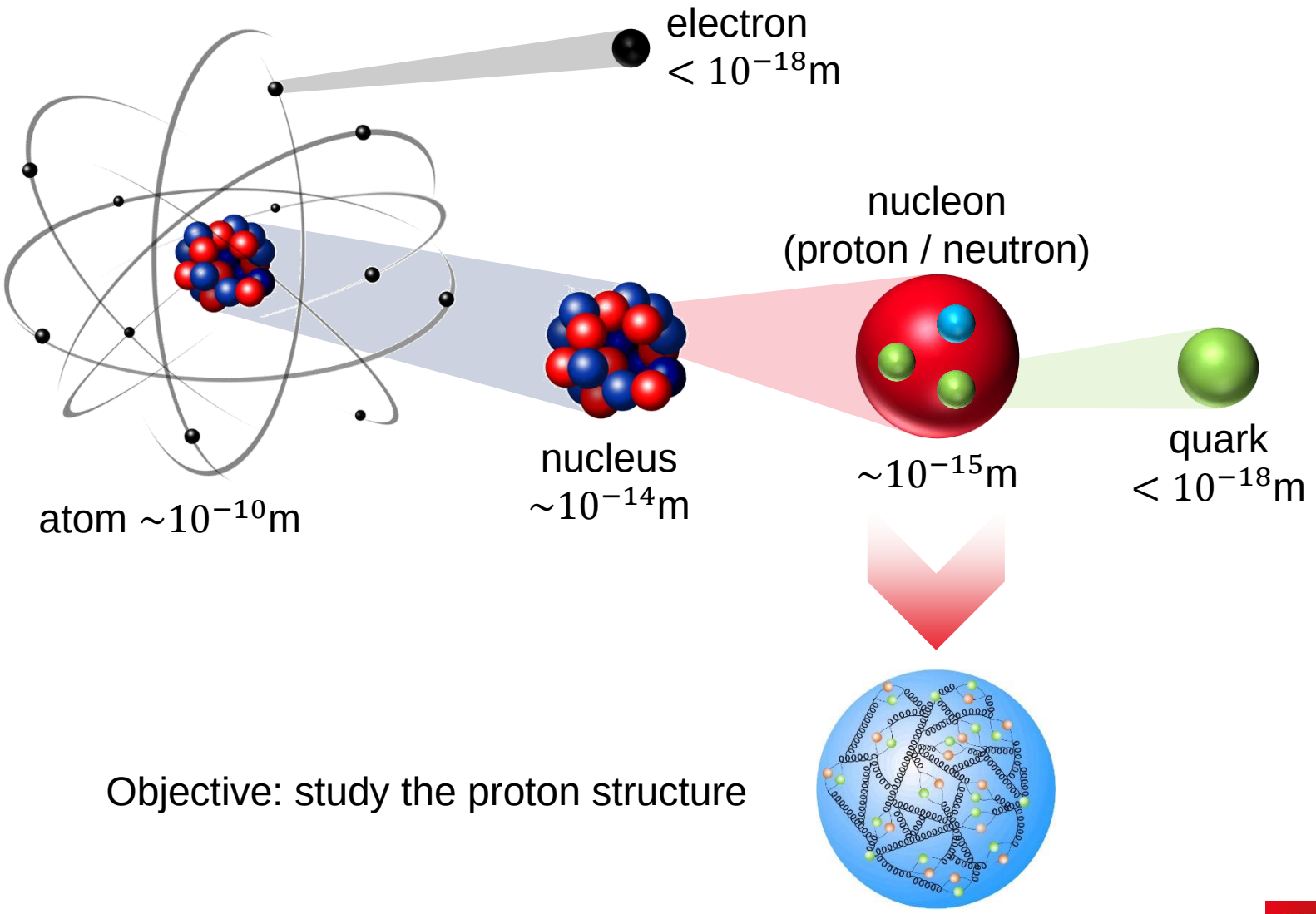
2. (Fuzzy) rule learning algorithms

3. Interpretable feature construction

OUTLINE

1. Introduction to the context and problem
2. (Fuzzy) rule learning algorithms
3. Interpretable feature construction

HADRONIC PHYSICS

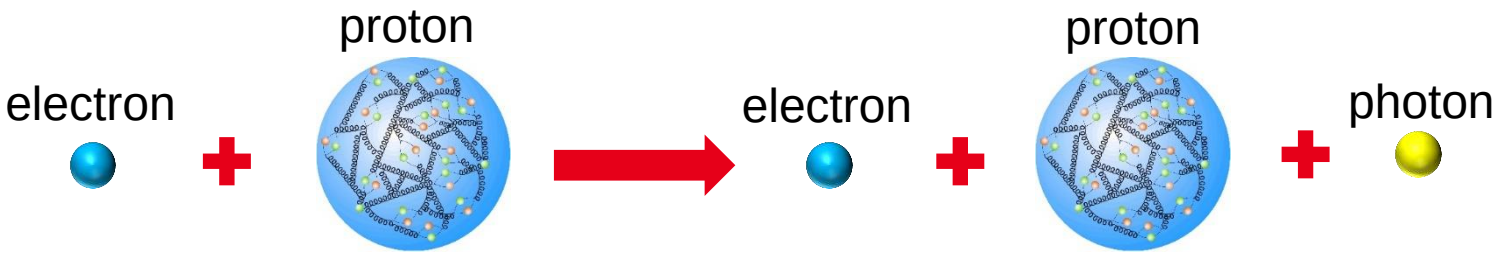


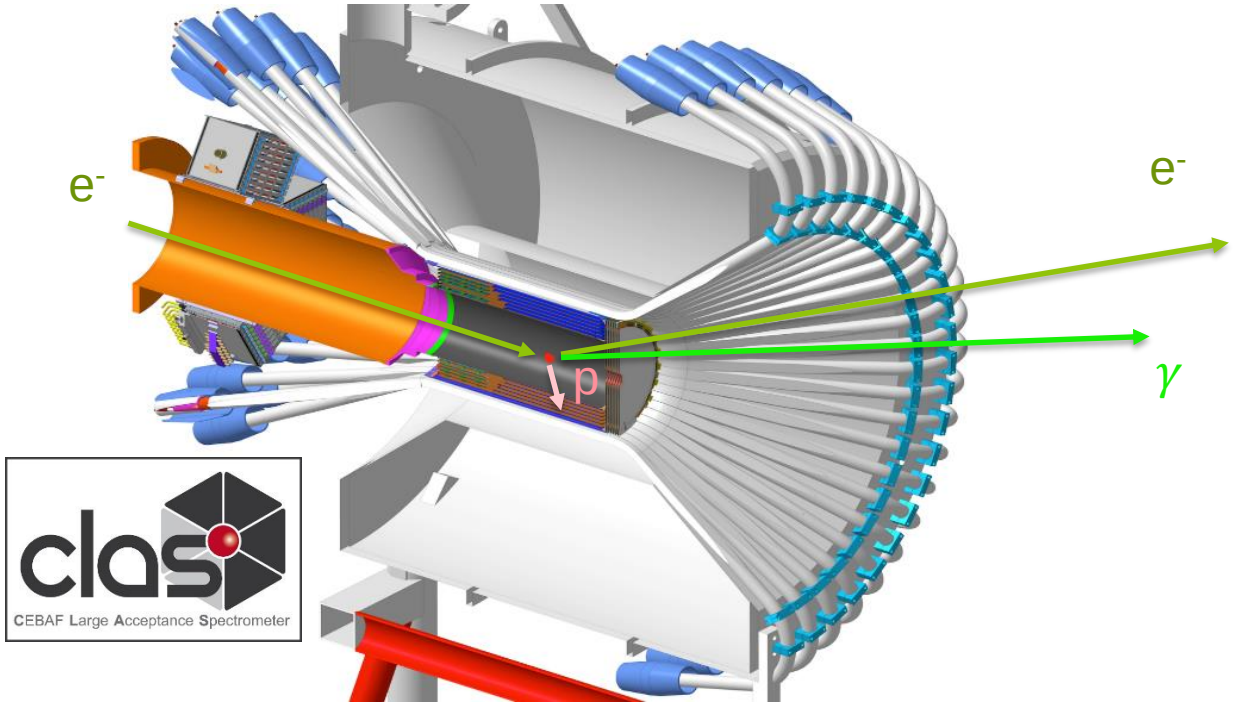
Objective: study the proton structure

OBJECTIVE IN EXPERIMENTAL PHYSICS

Study the proton structure

DVCS interaction (Deeply Virtual Compton Scattering)

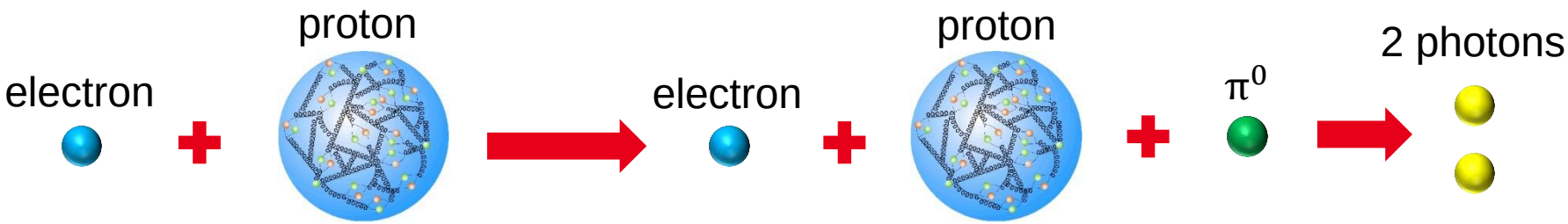


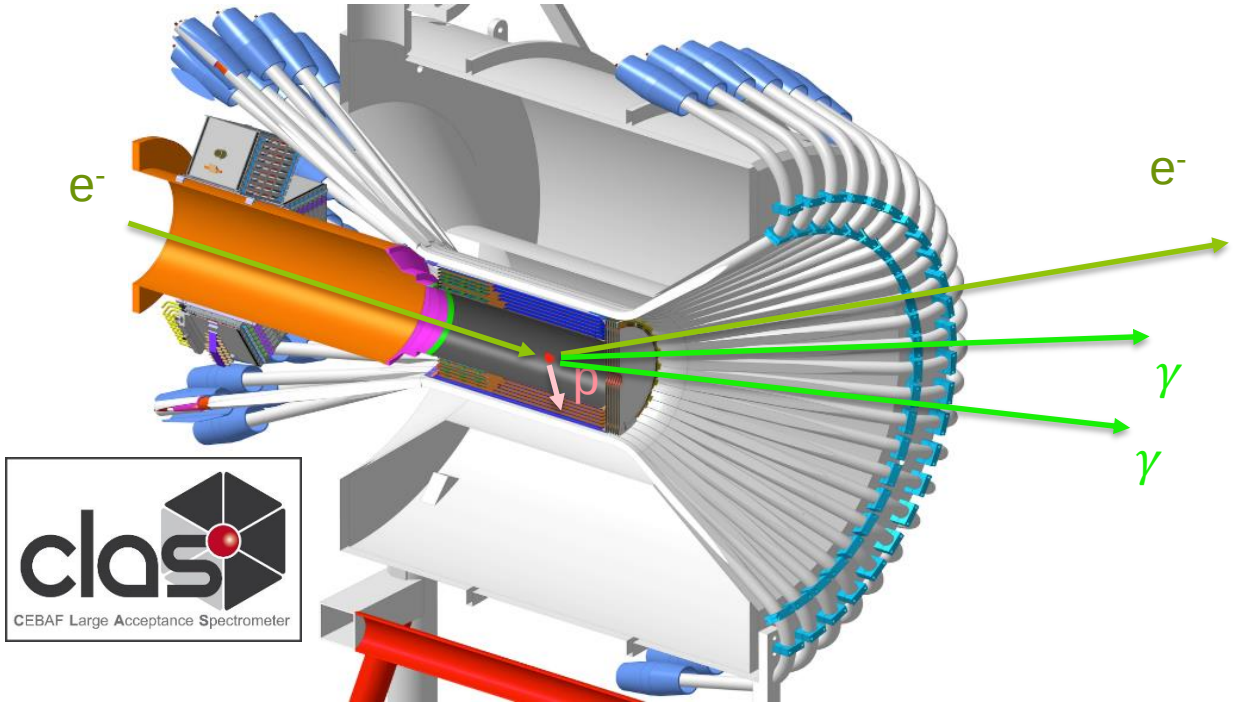


OBJECTIVE IN EXPERIMENTAL PHYSICS

Study the proton structure

Main background: Pi0 production





EXAMPLE IN EXPERIMENTAL PHYSICS

$$e^- + p \rightarrow e^- + p + \gamma$$

Conservation of energy and momentum

$$p_i = mv_i$$

Missing mass: $\|p_{e_{out}} + p_{p_{out}} + p_{\gamma_{out}} - p_{e_{in}} - p_{p_{in}}\| = 0$

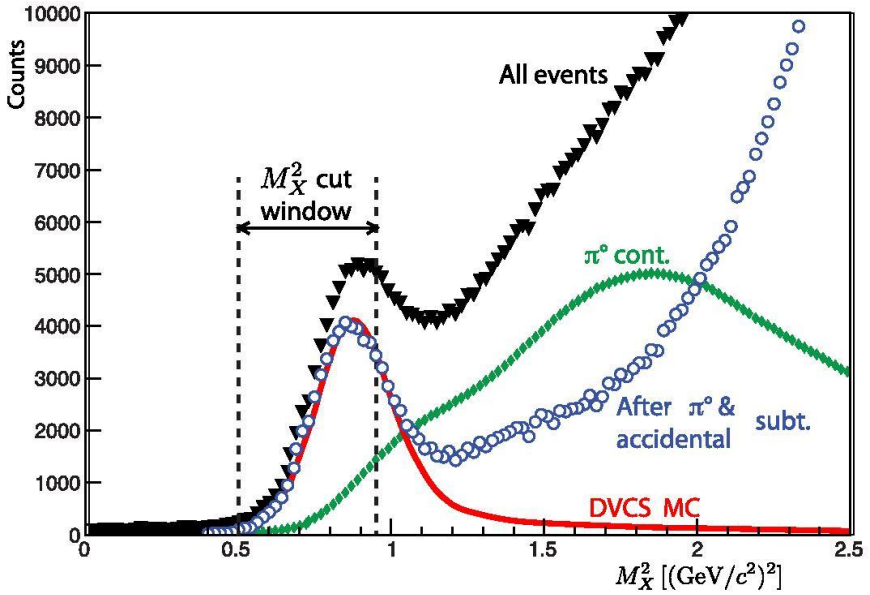
$$p = \begin{pmatrix} p_x \\ p_y \\ p_z \\ E \end{pmatrix}$$

Example of rule used by physicists:

if $E_\gamma > 3\ GeV$ and $\left|M_{ep\gamma}{}^2\right| < 0{,}5\ GeV^2$ and ..... then DVCS

Missing mass egX:

$$\|p_{e_{out}} + p_{\gamma_{out}} - p_{e_{in}} - p_{p_{in}}\| \sim M_p$$



→ Use of thresholds over complex high-level variables

OBJECTIVES OF THIS WORK

- Discriminate between DVCS (signal) events and π^0 (background) events in CLAS12 data

- Improve the analysis techniques with Machine Learning

- Use interpretable models

✓	Linear regression	$y = AX + b$
✓	K-Nearest Neighbors	$y =$ vote among the k NNs
✓	Decision trees	IF $x_1 > 2.7$ AND $x_2 < -4.3$ THEN $y =$ signal
≈	Boosted decision trees	$y =$ vote among trees
✗	Neural networks	$y =$ complex non-linear function of inputs

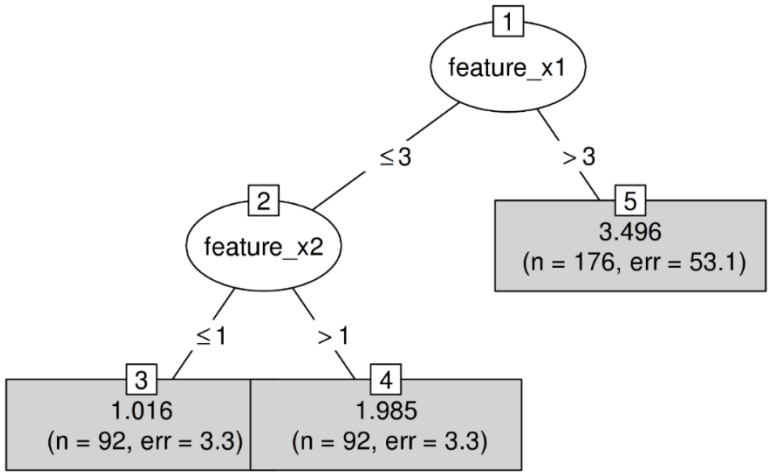
ORGANISATION

- Inputs from CLAS12 reconstructed data: list of identified particles and their momentum

Build list of rules

If $x_1 > \alpha_1$ and $x_2 < \alpha_2$ then DVCS
If $x_1 < \alpha_3$ and $x_3 > \alpha_4$ then Bkg
...

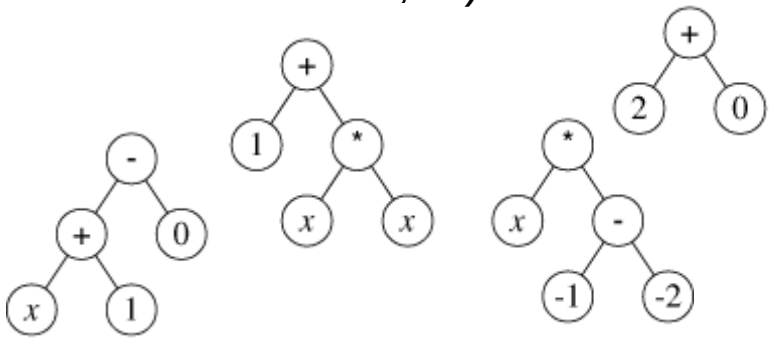
→ Decision trees, FURIA, ...



Extract relevant variables

$p_z^e + p_z^p + p_z^\gamma$
$\cos(\theta_{\gamma_1} - \theta_{\gamma_2})$
$\sqrt{E_e}$ ...

→ Feature construction with genetic programming (or other evolutionary methods, or tree search methods, ...)



OUTLINE

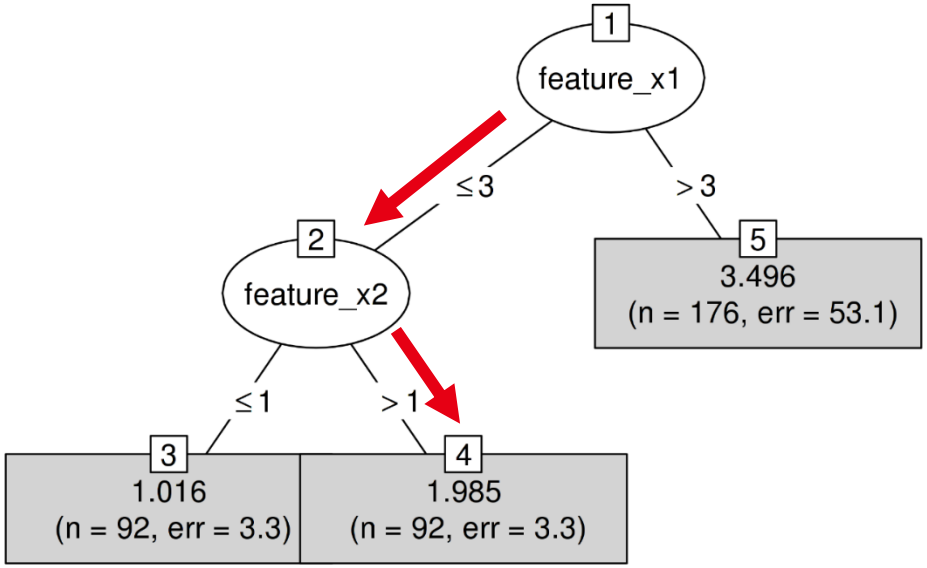
1. Introduction to the context and problem

2. (Fuzzy) rule learning algorithms

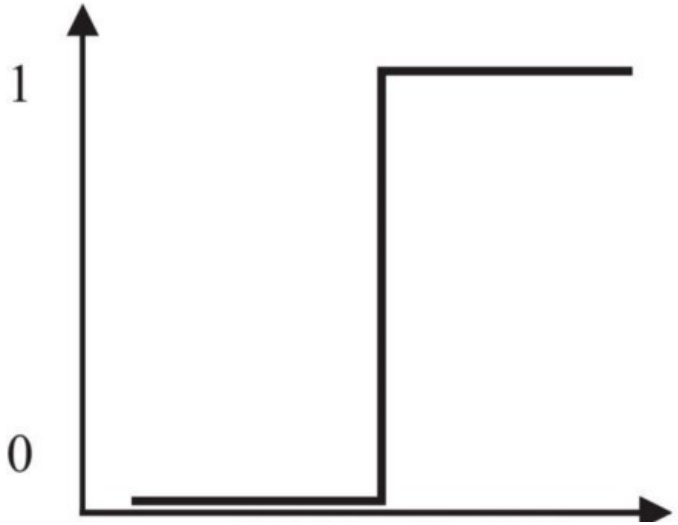
3. Interpretable feature construction

DECISION TREES

- At each junction, only one variable is used
- The variable which is used is the one optimizing a measure in the child nodes (e.g. min. entropy, max. information gain, ...)

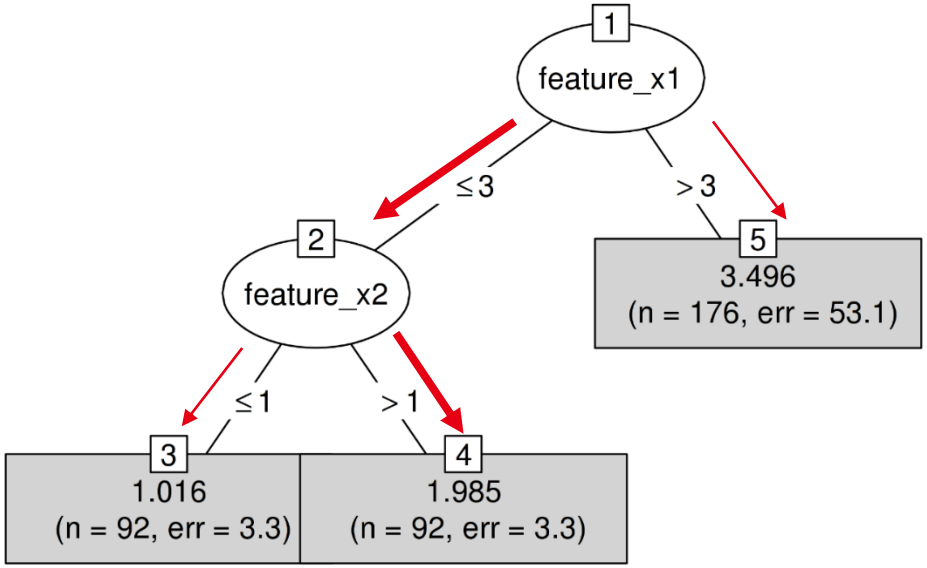


$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

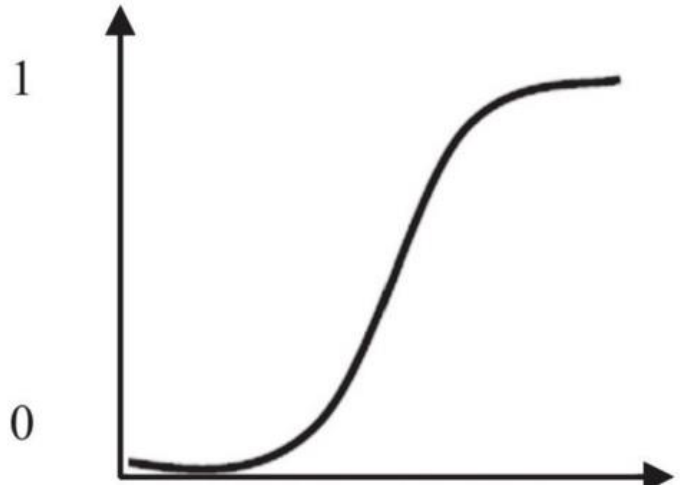


FUZZY DECISION TREES

- At each junction, only one variable is used
- The variable which is used is the one optimizing a measure in the child nodes (e.g. min. entropy, max. information gain, ...)

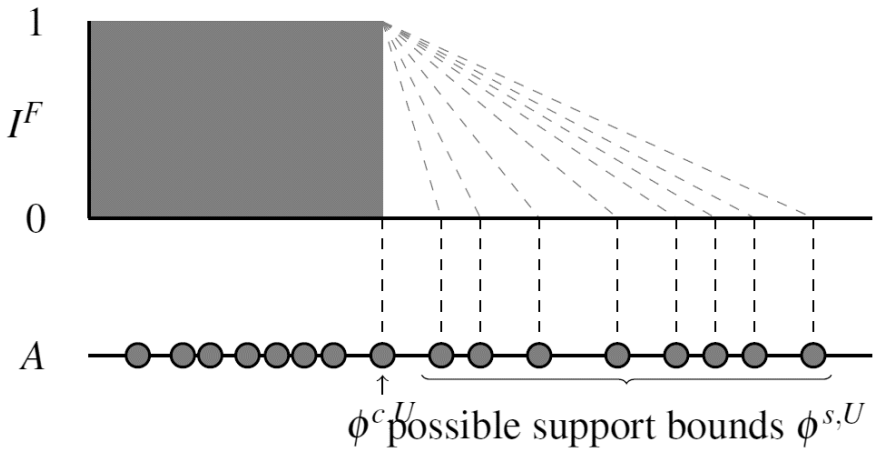


$$E = -\sum_{c_k} fr_{c_k/n_j} \log(fr_{c_k/n_j})$$



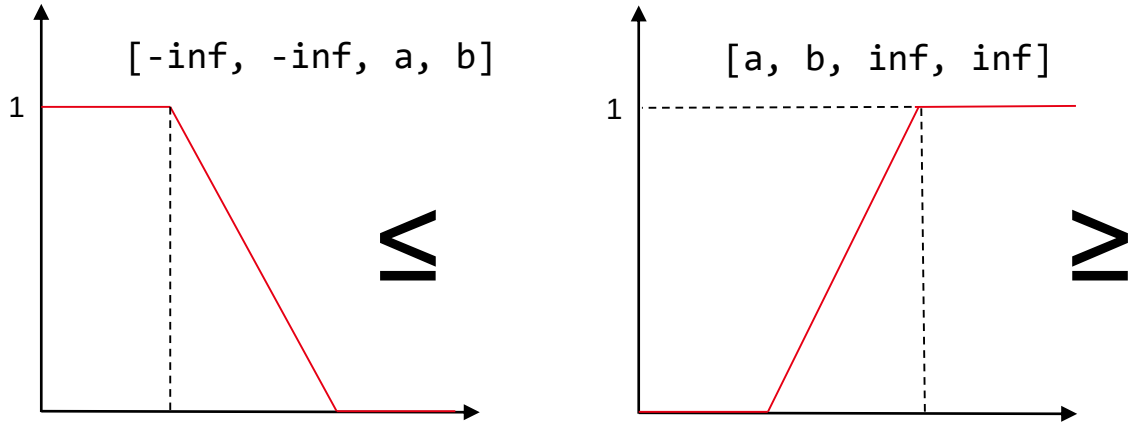
FURIA

- Extraction of a list of fuzzy rules
- Works generally better than decision trees, but worse than ensemble models (random forests, BDT...)
- Per class: add one rule to cover only instances from that class. Repeat until all instances of this class are covered by at least one rule (or stopping condition is reached)



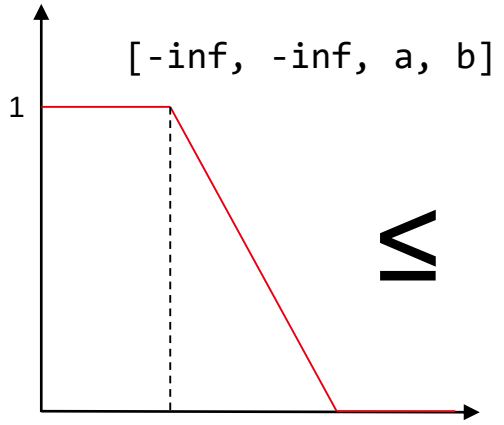
EXAMPLES OF RULES: FURIA

```
(inv_mass_g1g2 in [-inf, -inf, 0.665977, 0.666042]) and (inv_mass_g1g2 in [0.007705, 0.007706, inf,
inf]) => Class=DVMP (CF = 0.8)
(energy_g1 in [-inf, -inf, 2.209962, 2.21012]) and (cone_angle_g1 in [-inf, -inf, 16.272992,
16.275288]) => Class=DVMP (CF = 0.76)
(energy_g1 in [-inf, -inf, 3.100969, 3.101338]) and (MM_eg1 in [0.525376, 0.525439, inf, inf]) =>
Class=DVMP (CF = 0.65)
(energy_g1 in [-inf, -inf, 1.735166, 2.66702]) and (MM_eg1 in [-1.85998, -1.857006, inf, inf]) =>
Class=DVMP (CF = 0.61)
(MM_eg1 in [1.298545, 1.304201, inf, inf]) and (energy_g1 in [-inf, -inf, 4.182, 4.182101]) =>
Class=DVMP (CF = 0.66)
(energy_g1 in [3.333313, 3.333823, inf, inf]) and (MM_eg1 in [-inf, -inf, 0.96117, 0.961204]) =>
Class=DVCS (CF = 0.82)
(energy_g1 in [3.100909, 3.101237, inf, inf]) and (MM_eg1 in [-inf, -inf, 1.084021, 1.084045]) =>
Class=DVCS (CF = 0.8)
(MM_eg1 in [-inf, -inf, 0.852413, 0.852521]) and (energy_g1 in [2.103109, 2.103411, inf, inf]) =>
Class=DVCS (CF = 0.76)
(cone_angle_g1 in [16.137178, 21.604087, inf, inf]) and (MM_epg1 in [-inf, -inf, -0.538689, -
0.537701]) => Class=DVCS (CF = 0.56)
```

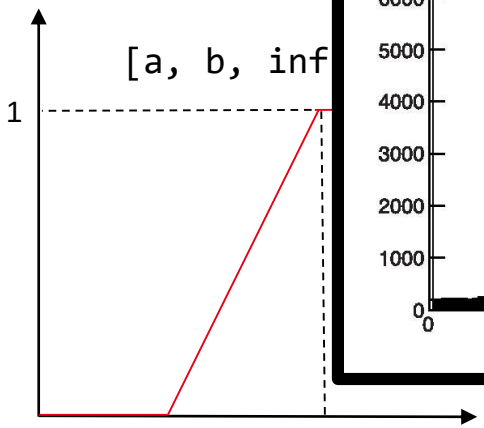


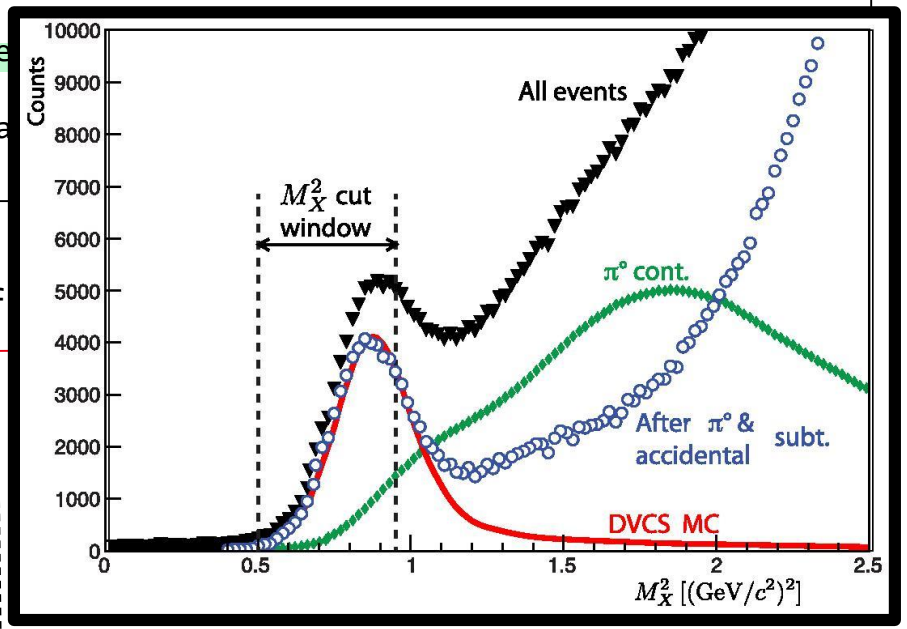
EXAMPLES OF RULES: FURIA

```
(inv_mass_g1g2 in [-inf, -inf, 0.665977, 0.666042]) and (inv_mass_g1g2 in [0.007705, 0.007706, inf,
inf]) => Class=DVMP (CF = 0.8)
(energy_g1 in [-inf, -inf, 2.209962, 2.21012]) and (cone_angle_g1 in [-inf, -inf, 16.272992,
16.275288]) => Class=DVMP (CF = 0.76)
(energy_g1 in [-inf, -inf, 3.100969, 3.101338]) and (MM_eg1 in [0.525376, 0.525439, inf, inf]) =>
Class=DVMP (CF = 0.65)
(energy_g1 in [-inf, -inf, 1.735166, 2.66702]) and (MM_eg1 in [-1.85998, -1.857006, inf, inf]) =>
Class=DVMP (CF = 0.61)
(MM_eg1 in [1.298545, 1.304201, inf, inf]) and (energy_g1 in [-inf, -inf, 4.182, 4.182101]) =>
Class=DVMP (CF = 0.66)
(energy_g1 in [3.333313, 3.333823, inf, inf]) and (MM_eg1 in [-inf, -inf, 0.96117, 0.961204]) =>
Class=DVCS (CF = 0.82)
(energy_g1 in [3.100909, 3.101237, inf, inf]) and (MM_eg1 in [-inf, -inf, 1.084021, 1.084045]) =>
Class=DVCS (CF = 0.8)
(MM_eg1 in [-inf, -inf, 0.852413, 0.852521]) and (ene
Class=DVCS (CF = 0.76)
(cone_angle_g1 in [16.137178, 21.604087, inf, inf]) a
0.537701]) => Class=DVCS (CF = 0.56)
```



≥





EXAMPLES OF RULES: DECISION TREE

pz_g1 <= 3.58181142807 :

pt_g1 <= 1.46455866001 :

pt_p <= 1.12264358222 :

pt_g2 <= 1.31972688944 :

(sin(phi_g2))*((py_g1)-((px_g1)+((pz_e)*((py_p)/(sqrt((px_p)*(pz_g1)))))) <= -1.63660074923 :

pt_g1 <= 1.3723567677 :

pt_g2 <= 0.155998215886 : DVMP (245.447503921/239.190329975)

pt_g2 > 0.155998215886 : DVMP (1403.22203666/1034.9474953)

pt_g1 > 1.3723567677 : DVCS (58.7913850369/35.7023099159)

(sin(phi_g2))*((py_g1)-((px_g1)+((pz_e)*((py_p)/(sqrt((px_p)*(pz_g1)))))) > -1.63660074923 :

pz_g1 <= 1.03863310814 :

pt_g2 <= 0.466542698848 : DVMP (1311.44090136/1086.07441752)

pt_g2 > 0.466542698848 : DVMP (453.914777635/312.193901424)

pz_g1 > 1.03863310814 :

square(pt_g1) <= 1.64576026784 :

pt_g2 <= 0.348592915007 : DVMP (1458.23311368/900.827944612)

pt_g2 > 0.348592915007 :

pz_e <= 6.23909330368 :

pt_g1 <= 0.711009710726 : DVMP (1078.04645148/640.142503317)

pt_g1 > 0.711009710726 :

(px_g1)*(px_e) <= 0.328979378184 : DVMP (1502.43430151/611.776665333)

(px_g1)*(px_e) > 0.328979378184 : DVCS (84.9173341296/66.9488340934)

pz_e > 6.23909330368 :

sqrt((py_p)/((pt_e)*(((phi_e)/(theta_g2))*((sin(phi_p))/((py_p)/(pt_e)))))) <= 0.120128532002 : DVMP (93.9918561927/67.5888460444)

sqrt((py_p)/((pt_e)*(((phi_e)/(theta_g2))*((sin(phi_p))/((py_p)/(pt_e)))))) > 0.120128532002 :

pz_g2 <= 1.35058891773 : DVMP (44.7036131905/32.1672115884)

pz_g2 > 1.35058891773 : DVCS (57.1762163768/55.6346527748)

square(pt_g1) > 1.64576026784 : DVMP (430.455063454/367.765620709)

pt_g2 > 1.31972688944 : DVCS (353.010491308/249.587489863)

pt_p > 1.12264358222 :

theta_g1 <= 10.1226023578 :

(px_g1)-((px_g1)-(((cos(phi_g1))*(px_g2))-((py_g1)+(sqrt((pz_g1)*(pt_p)))))) <= -2.10594519616 : DVMP (254.594427113/158.807825505)

(px_g1)-((px_g1)-(((cos(phi_g1))*(px_g2))-((py_g1)+(sqrt((pz_g1)*(pt_p)))))) > -2.10594519616 :

cos(theta_g1) <= 0.996798372318 :

theta_g2 <= 5.53926083661 : DVMP (83.3778071784/31.1781062461)

theta_g2 > 5.53926083661 : DVMP (746.485150183/424.665838695)

cos(theta_g1) > 0.996798372318 :

pz_g2 <= 1.12503802776 :

(theta_g1)-(theta_g2) <= -1.29802561943 : DVMP (179.949147082/63.3958740174)

(theta_g1)-(theta_g2) > -1.29802561943 : DVMP (30.9020716885/20.2533592264)

pz_g2 > 1.12503802776 :

(py_e)-((pt_g1)+((pz_g1)+(((theta_g1)/(theta_e))*((pz_e)*(square(sin(phi_p)))))) <= -1.59969202752 : DVMP (134.323689618/50.256755871)

(py_e)-((pt_g1)+((pz_g1)+(((theta_g1)/(theta_e))*((pz_e)*(square(sin(phi_p)))))) > -1.59969202752 : DVMP (90.6850620811/21.8597205916)

theta_g1 > 10.1226023578 :

pt_g2 <= 0.953768620235 :

pz_g1 <= 2.43110609055 :

(px_g2)/((cos(phi_p))/((px_g1)/(((px_g1)/(pt_p))*((pz_e)/(cos(theta_e)))))) <= 0.0292153916493 :

pt_g1 <= 1.05827961345 : DVMP (4005.52912589/2800.96030614)

pt_g1 > 1.05827961345 : DVCS (209.685441384/169.894368795)

(px_g2)/((cos(phi_p))/((px_g1)/(((px_g1)/(pt_p))*((pz_e)/(cos(theta_e)))))) > 0.0292153916493 :

pt_g1 <= 1.05223902879 :

(px_g3)*((pz_g3)/((sin(phi_g2))-((pz_g3)/((py_e)-((pt_g2)*(tan(phi_g1))))))) <= 0.211012725393 :

(py_g1)*(((phi_g3)/((phi_g1)-(theta_p)))*(((pt_e)/(pt_g3))*((py_p)/(pt_e)))) <= -1.36452838141 : DVMP (349.603273408/262.291035742)

(py_g1)*(((phi_g3)/((phi_g1)-(theta_p)))*(((pt_e)/(pt_g3))*((py_p)/(pt_e)))) > -1.36452838141 : DVMP (4924.37778012/4452.17406199)

(px_g3)*((pz_g3)/((sin(phi_g2))-((pz_g3)/((py_e)-((pt_g2)*(tan(phi_g1))))))) > 0.211012725393 : DVMP (578.720502583/445.560451064)

pt_g1 > 1.05223902879 : DVCS (401.433990228/276.896846899)

pz_g1 > 2.43110609055 :

pt_g1 <= 1.03497340619 :

(pz_g3)-((pt_g3)-((pz_p)-(((pz_g2)-((py_e)-(pz_g2)))-((pz_e)-(pt_g1))))) <= 3.53693377864 :

sin(theta_g2) <= 0.383863787973 : DVMP (497.012033798/411.112343833)

sin(theta_g2) > 0.383863787973 : DVCS (118.088260272/86.2400524372)

(pz_g3)-((pt_g3)-((pz_p)-(((pz_g2)-((py_e)-(pz_g2)))-((pz_e)-(pt_g1))))) > 3.53693377864 :

theta_g2 <= 20.7547126474 : DVMP (489.746363828/457.324010133)

theta_g2 > 20.7547126474 : DVCS (260.118129516/192.967468848)

pt_g1 > 1.03497340619 :

(py_g2)/(py_g1) <= 0.114276902752 : DVCS (535.694385618/291.303454651)

(py_g2)/(py_g1) > 0.114276902752 : DVCS (530.229613322/511.595319489)

pt_g2 > 0.953768620235 : DVCS (747.111476101/511.903322701)

pt_g1 > 1.46455866001 :

pt_g1 <= 1.67138147249 : DVCS (1017.63714713/576.729553249)

pt_g1 > 1.67138147249 : DVCS (780.349221704/211.284341747)

pz_g1 > 3.58181142807 :

pt_g1 <= 1.67343862638 :

pz_g1 <= 4.83315992355 :

theta_g2 <= 21.1505417079 :

pz_g2 <= 4.28238010406 :

pt_g1 <= 1.4162697332 :

square(pz_p) <= 4.44053461214 :

pz_e <= 4.39802646637 :

theta_g2 <= 17.1600816375 : DVMP (452.154600736/171.315418495)

theta_g2 > 17.1600816375 : DVMP (94.412633905/55.5906918095)

pz_e > 4.39802646637 :

pz_g2 <= 2.16436076164 :

pt_p <= 0.863928199945 :

pz_e <= 5.3938794136 :

pz_g1 <= 4.28205060959 : DVMP (52.6739013395/18.8194770775)

pz_g1 > 4.28205060959 :

pz_g2 <= 1.03894972801 :

pz_g2 <= 0.593278586864 : DVCS (3.564776197/0.900646023412)

pz_g2 > 0.593278586864 : DVCS (13.937841838/12.8546914845)

pz_g2 > 1.03894972801 : DVCS (7.03542455283/1.77751050735)

pz_e > 5.3938794136 : DVCS (48.4050293236/16.9024030334)

pt_p > 0.863928199945 : DVCS (93.4649119673/26.4495477603)

pz_g2 > 2.16436076164 : DVCS (89.3724814483/33.0421815462)

square(pz_p) > 4.44053461214 : DVCS (937.287796707/809.552696875)

pt_g1 > 1.4162697332 : DVCS (527.203202088/338.580010767)

pz_g2 > 4.28238010406 : DVCS (105.320251431/42.7119313676)

theta_g2 > 21.1505417079 : DVCS (1089.84301718/686.543176411)

pz_g1 > 4.83315992355 :

pz_p <= 4.24705791473 :

pz_e <= 2.50621247292 :

pz_g1 <= 6.97984361649 :

theta_g2 <= 17.8138623973 :

pz_p <= 2.37969708443 :

theta_g3 <= 21.7494735716 :

pz_g1 <= 5.64450550079 : DVMP (78.7387628683/25.0505226951)

pz_g1 > 5.64450550079 :

(pz_g2)*((px_g2)/((px_g1)/(((pt_p)-(px_p))+((px_p)/((py_p)/(py_e)))))) <= 1.74143903284 : DVMP (69.7147043258/55.6532569283)

(pz_g2)*((px_g2)/((px_g1)/(((pt_p)-(px_p))+((px_p)/((py_p)/(py_e)))))) > 1.74143903284 : DVCS (19.4134673201/12.0977899382)

theta_g3 > 17.7494735716 : DVMP (39.9782549423/35.4394422501)

pz_p > 2.37969708443 : DVCS (176.057171129/80.4521341679)

theta_g2 > 17.8138623973 :

pz_p <= 2.39342975616 :

(pt_g2)/((py_p)*((py_e)/((sin(phi_e))/(((pt_g2)-(px_e))*(cos(theta_p)))))) <= -1.25052874809 : DVMP (25.2061137061/21.9222774161)

(pt_g2)/((py_p)*((py_e)/((sin(phi_e))/(((pt_g2)-(px_e))*(cos(theta_p)))))) > -1.25052874809 : DVCS (80.7927397763/44.8829698829)

pz_p > 2.39342975616 : DVCS (158.945071612/42.6454294808)

pz_g1 > 6.97984361649 :

pt_g2 <= 0.197378619762 : DVCS (17.6029922824/4.95122090394)

pt_g2 > 0.197378619762 : DVCS (187.889865232/22.4546795937)

pz_e > 2.50621247292 :

pt_g2 <= 0.2309034739 : DVCS (434.602972619/212.921982667)

pt_g2 > 0.2309034739 :

pz_e <= 5.49648523331 :

theta_g3 <= 21.1998595157 :

pt_g2 <= 0.454606900353 :

(py_g2)*((py_g1)+((px_g1)*((sin(phi_e))-(sqrt((pz_g2)/((pt_e)+(px_g2))))))) <= 0.0481618185923 : DVCS (192.989228638/56.5339422084)

(py_g2)*((py_g1)+((px_g1)*((sin(phi_e))-(sqrt((pz_g2)/((pt_e)+(px_g2))))))) > 0.0481618185923 : DVCS (119.008599768/55.3679543883)

pt_g2 > 0.454606900353 : DVCS (380.485616517/100.807017983)

theta_g3 > 21.1998595157 : DVCS (378.509601302/100.47384968)

pz_e > 5.49648523331 : DVCS (49.1415055118/41.1789514852)

pz_p > 4.24705791473 : DVCS (281.041355398/206.021864679)

pt_g1 > 1.67343862638 :

pz_p <= 5.02861642838 :

pz_e <= 5.71568346024 :

pt_p <= 1.27249384286 :

pt_g1 <= 1.89532162759 : DVCS (1365.30295174/306.322208383)

pt_g1 > 1.89532162759 : DVCS (860.57303902/83.3675406258)

pt_p > 1.27249384286 : DVCS (385.366821087/121.202455962)

pz_e > 5.71568346024 :

square(pz_p) <= 2.36406285068 : DVCS (30.0557977423/5.86333445855)

square(pz_p) > 2.36406285068 : DVCS (57.9443306768/56.3322253211)

pz_p > 5.02861642838 : DVCS (252.293336226/143.04461147)

SUMMARY

- Rule learning algorithms are interpretable and suitable for physics analyses
- FURIA VS Decision Trees: less rules, similar performances
- Advantage of fuzzification: incorporate data imprecisions

Problem:

These algorithms highly rely on the input variables they use

OUTLINE

1. Introduction to the context and problem

2. (Fuzzy) rule learning algorithms

3. Interpretable feature construction

PRINCIPLE OF FEATURE CONSTRUCTION

- List of initial variables/features: $f_1, f_2, f_3, \ldots$

- Construction of one or several new feature(s) $f'_1 = \frac{f_1 + f_3}{f_2}$
  from a list of operators: $+,\ -,\ \times,\ \div,\ \sqrt{\ },\ \cos\ ,\ \log\ ,\ldots$

- Evaluation of the built feature(s):
  → **Filter** methods: measure independent of any ML algorithm (ex: entropy)

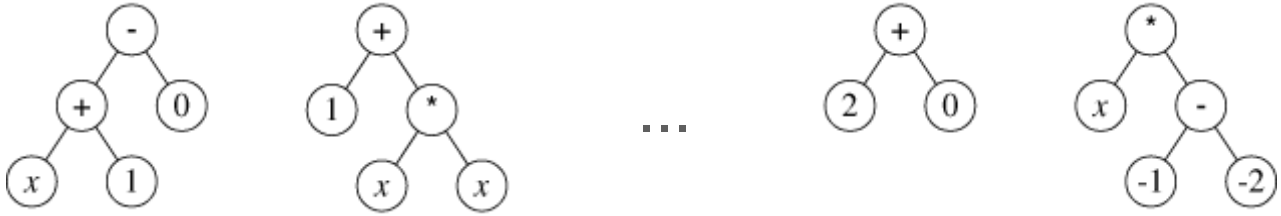
  → **Wrapper** methods: score of a trained classifier over a temporary test sample

  ex: accuracy $s = \frac{nb\ of\ correctly\ classified\ events}{total\ nb\ of\ events}$

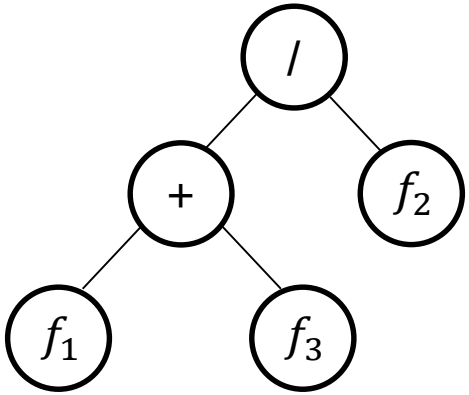
- Additional constraint: obtain physically sound features
  → Choice: Genetic Programming (do not add energies with angles)

CONSTRAINED GENETIC PROGRAMMING

- Evolve a population of **N** tree-like individuals



$$f'_1 = \frac{f_1 + f_3}{f_2} =$$



- At each generation, perform mutation and crossover between individuals
- Evaluate and select the best individuals to form the next generation
- The quality of the population is improved at each generation

GRAMMAR TO CONSTRAIN GENETIC PROGRAMMING

Grammar to authorize only valid operations (ex. do not add an energy and an angle)

```
<start> ::= <E> | <A> | <F>
<E> ::= <E> + <E> | <E> - <E> | <E> * <F>
        | <E> / <F> | sqrt(<E2>) | <termE>
<A> ::= <A> + <A> | <A> - <A> | acos(<F>)
        | asin(<F>) | atan(<F>) | <termA>
<F> ::= <F> + <F> | <F> - <F> | <F> * <F>
        | <F> / <F> | <E> / <E> | <A> / <A>
        | cos(<A>) | sin(<A>) | tan(<A>)
        | <termF>
<E2> ::= <E2> + <E2> | <E2> - <E2>
        | <E> * <E> | <E2> * <F> | <E2> / <F>
        | square(<E>) | <termE2>
```

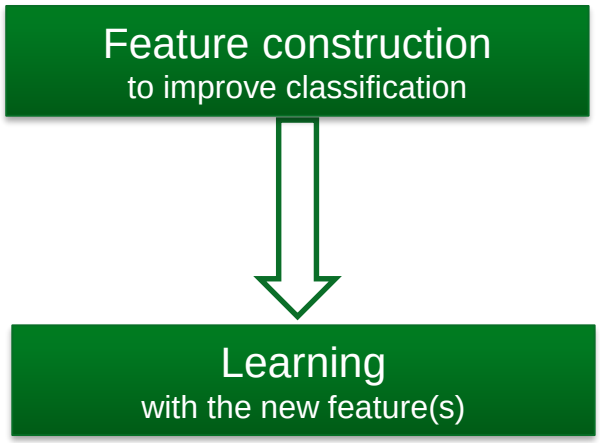
EXAMPLES OF BUILT FEATURES

Wrapper method: features are evaluated through a XGBoost training with and without this new feature

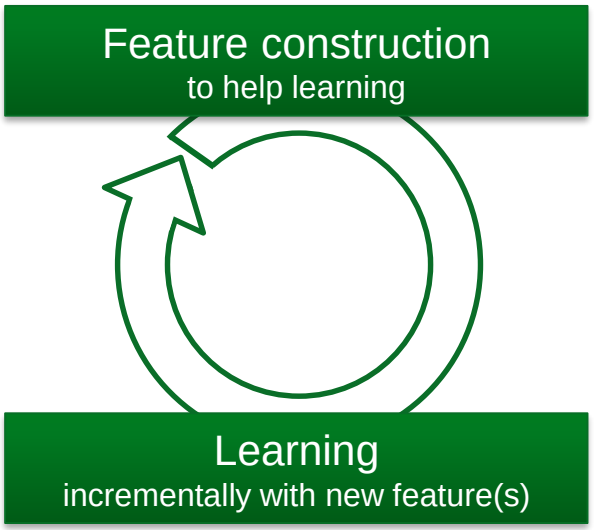
$p_z^e + p_z^{\gamma_1} + p_z^p$	+2.0%
$(\cos(\phi^{\gamma_1} - \phi^{\gamma_2}) + 9)\cos(\theta^{\gamma_1} - \theta^{\gamma_2})$	+1.6%
$\phi^{\gamma_1} - \phi^{\gamma_2}$	+1.0%

- Automatic feature construction may greatly improve the classification score, while keeping the interpretability/readability of newly built features
- The method is generic and can be used in any analysis

PROSPECTS



Merge the two methods?
Gain in time!



- Assess the performances of these different methods
  - ⟶ Preliminary VS embedded
  - ⟶ Number of features? Trade-off speed/performance

- Compare all these methods (goal: extract asymmetries and cross sections) to:
  - ⟶ Current physicists methods (1 PhD)
  - ⟶ « Black box » machine learning techniques and transfer learning (1 post-doc)