Interpretable machine learning algorithms for DVCS analysis in CLAS12 data

CLAS12 collaboration meeting
November 2018

Noëlie Cherrier (CEA Saclay)

OVERVIEW

1. Introduction to the problem we want to solve

2. Feature construction
   Genetic Programming
   Mixing both

3. Application to DVCS

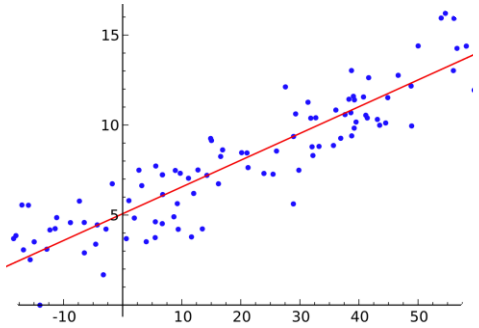
OBJECTIVES

- **Discriminate between DVCS (signal) events and background events in CLAS12 data**

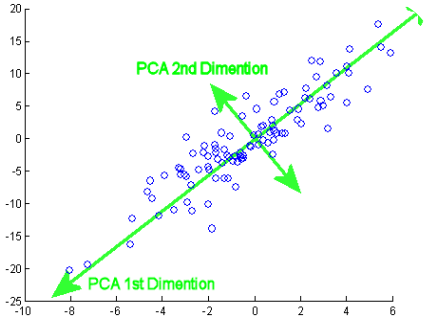
- **Improve the analysis techniques with Machine Learning**

MACHINE LEARNING: BASIC ALGORITHMS

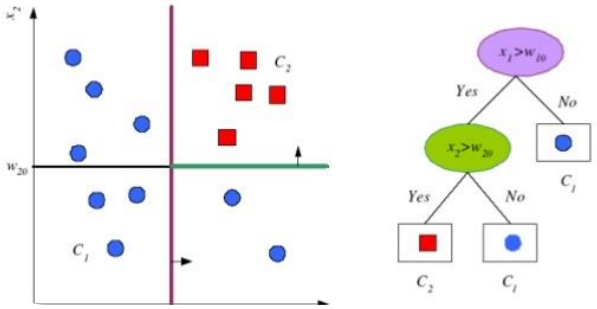
ML = statistical learning = extract knowledge from data



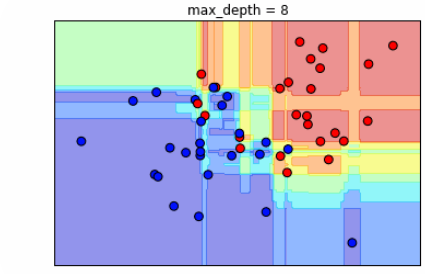
Linear regression



Principal Component Analysis



Decision Tree



Boosted Decision Trees

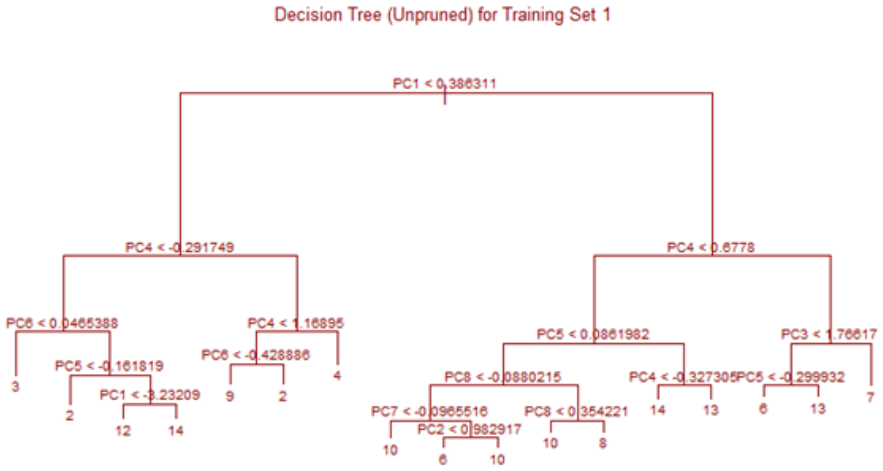
OBJECTIVES

- **Discriminate between DVCS (signal) events and 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 my 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

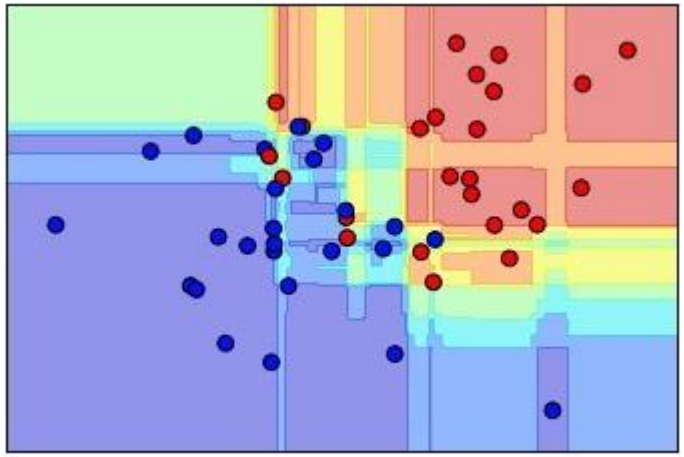
DOWNSIDES OF DECISION TREES

- At each junction, only one variable is used



- Output: hyperrectangles

  (even for BDTs)



SOLUTION: USE OF HIGH-LEVEL VARIABLES

- **Output of reconstruction: list of particles and their 3-momentum, vertex, charge, etc.**

- **Variables used in cuts to select DVCS events: missing mass, cone angle**

  → Functions of available variables (momenta...)

SOLUTION: USE OF HIGH-LEVEL VARIABLES

- **Output of reconstruction: list of particles and their 3-momentum, vertex, charge, etc.**

- **Variables used in cuts to select DVCS events: missing mass, cone angle**

  → Functions of available variables (momenta...)

- **Idea: build automatically new variables to improve the score of a machine learning classifier**

  → **Feature construction**

SOLUTION: USE OF HIGH-LEVEL VARIABLES

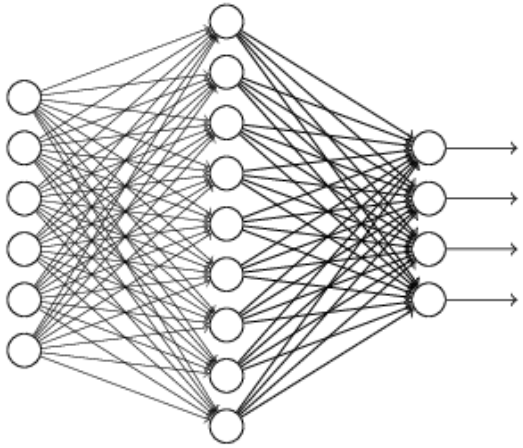
- **Output of reconstruction: list of particles and their 3-momentum, vertex, charge, etc.**

- **Variables used in cuts to select DVCS events: missing mass, cone angle**

  → Functions of available variables (momenta...)

- **Idea: build automatically new variables to improve the score of a machine learning classifier**

  → **Feature construction**



SOLUTION: USE OF HIGH-LEVEL VARIABLES

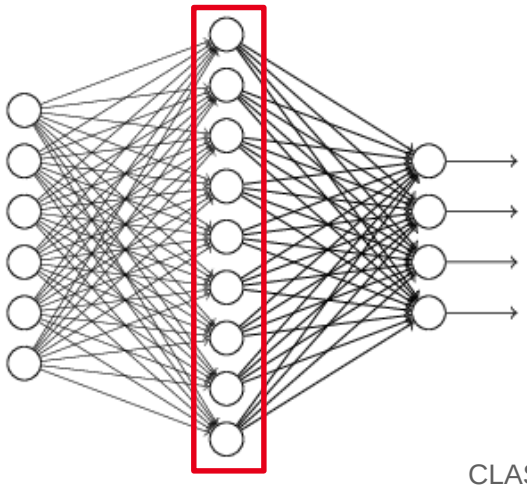
- **Output of reconstruction: list of particles and their 3-momentum, vertex, charge, etc.**

- **Variables used in cuts to select DVCS events: missing mass, cone angle**

  → Functions of available variables (momenta...)

- **Idea: build automatically new variables to improve the score of a machine learning classifier**

  → **Feature construction**



SOLUTION: USE OF HIGH-LEVEL VARIABLES

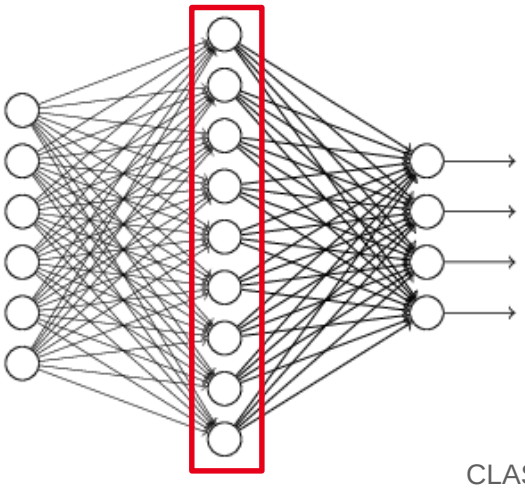
- **Output of reconstruction: list of particles and their 3-momentum, vertex, charge, etc.**

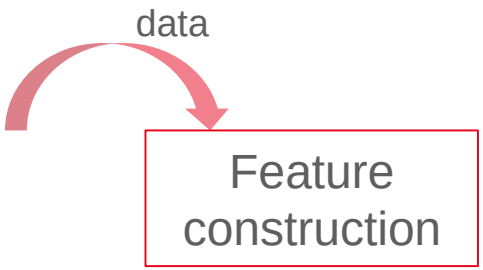
- **Variables used in cuts to select DVCS events: missing mass, cone angle**

  → Functions of available variables (momenta...)

- **Idea: build automatically new variables to improve the score of a machine learning classifier**

  → **Feature construction**





SOLUTION: USE OF HIGH-LEVEL VARIABLES

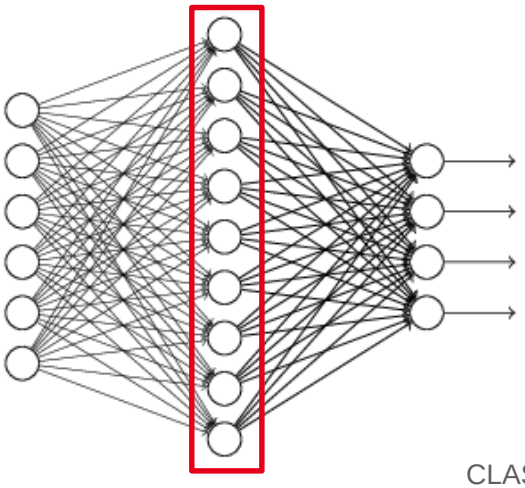
- **Output of reconstruction: list of particles and their 3-momentum, vertex, charge, etc.**

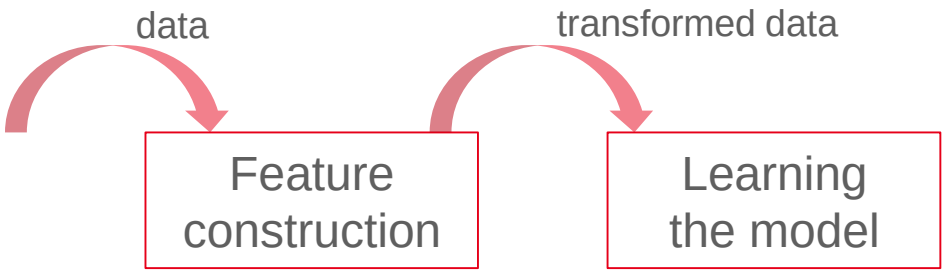
- **Variables used in cuts to select DVCS events: missing mass, cone angle**

  → Functions of available variables (momenta...)

- **Idea: build automatically new variables to improve the score of a machine learning classifier**

  → **Feature construction**





FEATURE CONSTRUCTION

Principle

- **List of initial variables/features: f_1, f_2, f_3, ...**

- **Construction of a new feature $f'_1 = \frac{f_1 + f_3}{f_2}$**

- **Construction of one or several new features**
- **Addition of the new features to the initial list or replacement of the list with the new features**

- **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 $s = \frac{nb\ of\ correctly\ classified\ events}{total\ nb\ of\ events}$

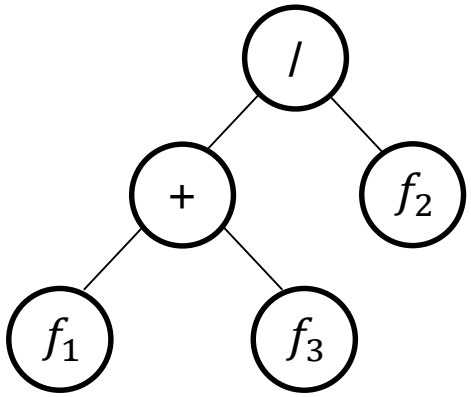
FEATURE CONSTRUCTION

Principle

- **Explicit construction of numeric features from a list of operators:**
  $+,\ -,\ \times,\ \div,\ \sqrt{\ },\ \cos\ ,\ \log\ ,$ **...**

- **Tree representation:**

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



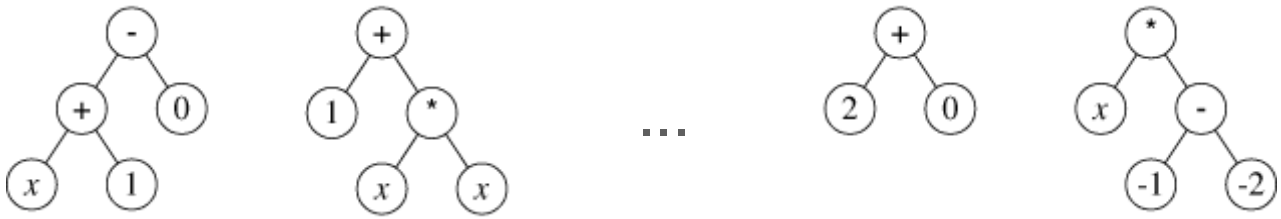
- **Numerous methods exist in the literature**

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

GENETIC PROGRAMMING

Principle

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



- **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**

GENETIC PROGRAMMING

Constraints: how to ensure interpretability

- **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>
```

GENETIC PROGRAMMING

Constraints: how to ensure interpretability

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

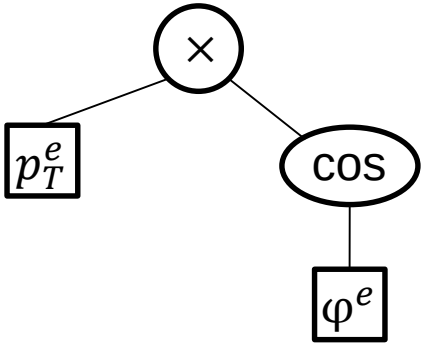
- **Transition probabilities to reproduce frequent patterns in physics formulas**

		Energy			Energy^2		
		$+$	$-$	$\sqrt{}$	$+$	$-$	^{2}
Energy	$+$	0.1	0.225				0.8
	$-$	0.1	0.225				0.1
	$\times$	0.1	0.25				0.07
	$\div$	0.1	0.2				0.03
	$\sqrt{}$	0.6	0.1				0
Energy^2	$+$			0.7	0.4	0.2	
	$-$			0.25	0.15	0.07	
	$\times$			0.05	0.05	0.03	
	^{2}			0	0.4	0.7	

GENETIC PROGRAMMING

Evolution details

- **Tree-like individuals are generated following the grammar and transition probabilities**



- **Crossover and mutation are performed at each generation on certain individuals (altering the branches)**
- **The newly created individuals are evaluated (XGBoost on the dataset augmented with the newly built feature)**
- **Tournament selection: select randomly 3 individuals and keep the best (repeat until the new population is full)**
- **Repeat for 150 generations of 500 individuals each**

APPLICATION TO CLAS12 DATA

Data

- **Need for labeled data (which event is DVCS, which event is background)**
- **MC data: uniform generation of DVCS and π^0 production events**
- **Base features: one electron, one proton, three photons**

  Selection: at least 2 of these particles including an electron

  $E > 2\ GeV$

  $Q^2 > 1.5\ GeV^2$

  $W^2 > 4\ GeV^2$

- **3-momentum for each particle**
- **Base score with XGBoost: 64% of correctly classified MC events**

APPLICATION TO CLAS12 DATA

Examples

- **Examples of high-level features that can be used for cuts:**
  - → M[ep→epγ] +0.77%
  - → M[ep→epX] +0.23%
  - → Photon energy +0.23%
  - → Cone angle +1.69%
  - → Photon energy + cone angle +1.75%
  - → All at once +2.04%

- **2 examples of feature built by the feature construction algorithm (from 2 independent runs, vectorial representation with operators such as norm, angle, dot product, ...):**

$$M_p + (\vec{p_e} + \vec{p_p} + \vec{p_{\gamma_1}})_z$$

$$\|\vec{p_e} + \vec{p_p} + \vec{p_{\gamma_1}}\|$$

Both add **2%** to the score, which is big compared to the examples above!

APPLICATION TO CLAS12 DATA

Examples

- **Other tests: with cartesian coordinates**

$$p_z^e + p_z^{\gamma_1} + p_z^p \qquad +2.0\%$$

- **With spherical coordinates**

$$\left(\cos\left(\phi^{\gamma_1} - \phi^{\gamma_2}\right) + 9\right)\cos\left(\theta^{\gamma_1} - \theta^{\gamma_2}\right) \qquad +1.6\%$$

$$\phi^{\gamma_1} - \phi^{\gamma_2} \qquad +1.0\%$$

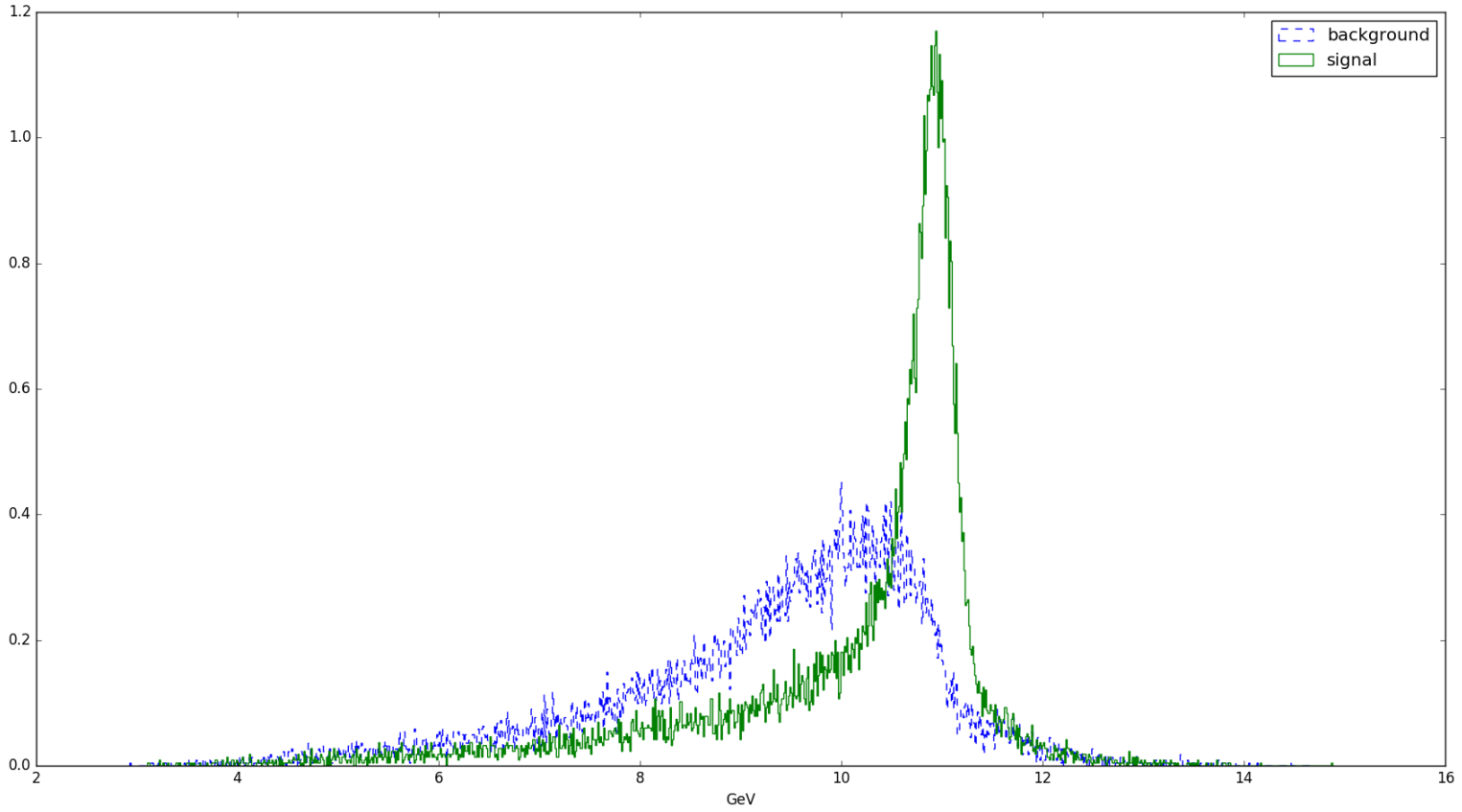
Mean over 15 runs:

Vectorial	+1.87 ± 0.24 %
Spherical	+0.98 ± 0.28 %
Cartesian	+1.93 ± 0.09 %

APPLICATION TO CLAS12 DATA

Discriminating power

$$\left\| \vec{p_e} + \vec{p_p} + \vec{p_{\gamma_1}} \right\|$$



SUMMARY

- 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

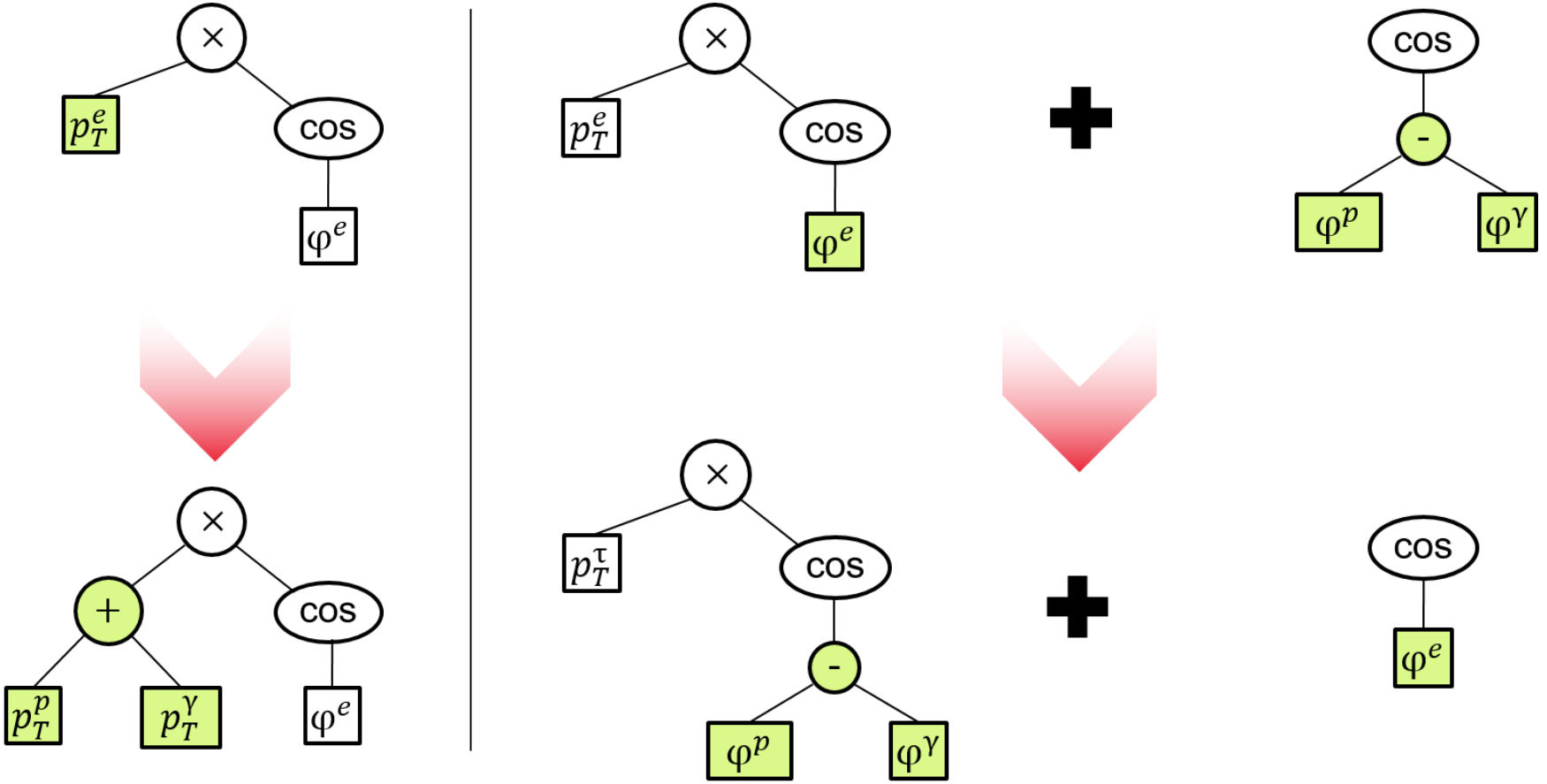
Prospects

- Merge all possibilities (cartesian, spherical coordinates or vectorized implementation)
- Use other learning algorithms
- Retro-engineering of the built features
- Test the algorithm on smaller regions of the phase space (same built features?)

GENETIC PROGRAMMING

Evolution details

- **At each generation: for each individual, mutation and/or crossover can occur with a certain probability**



GENETIC PROGRAMMING

Evolution details

- **The newly created individuals are evaluated:**

  3-fold cross-validation on the train set with a ML model (here XGBoost)
  The score is the mean accuracy over the 3 folds
  The data is the base dataset with the newly built feature(s) added to the initial list of variables

- **A selection is performed to get the next generation of individuals:**

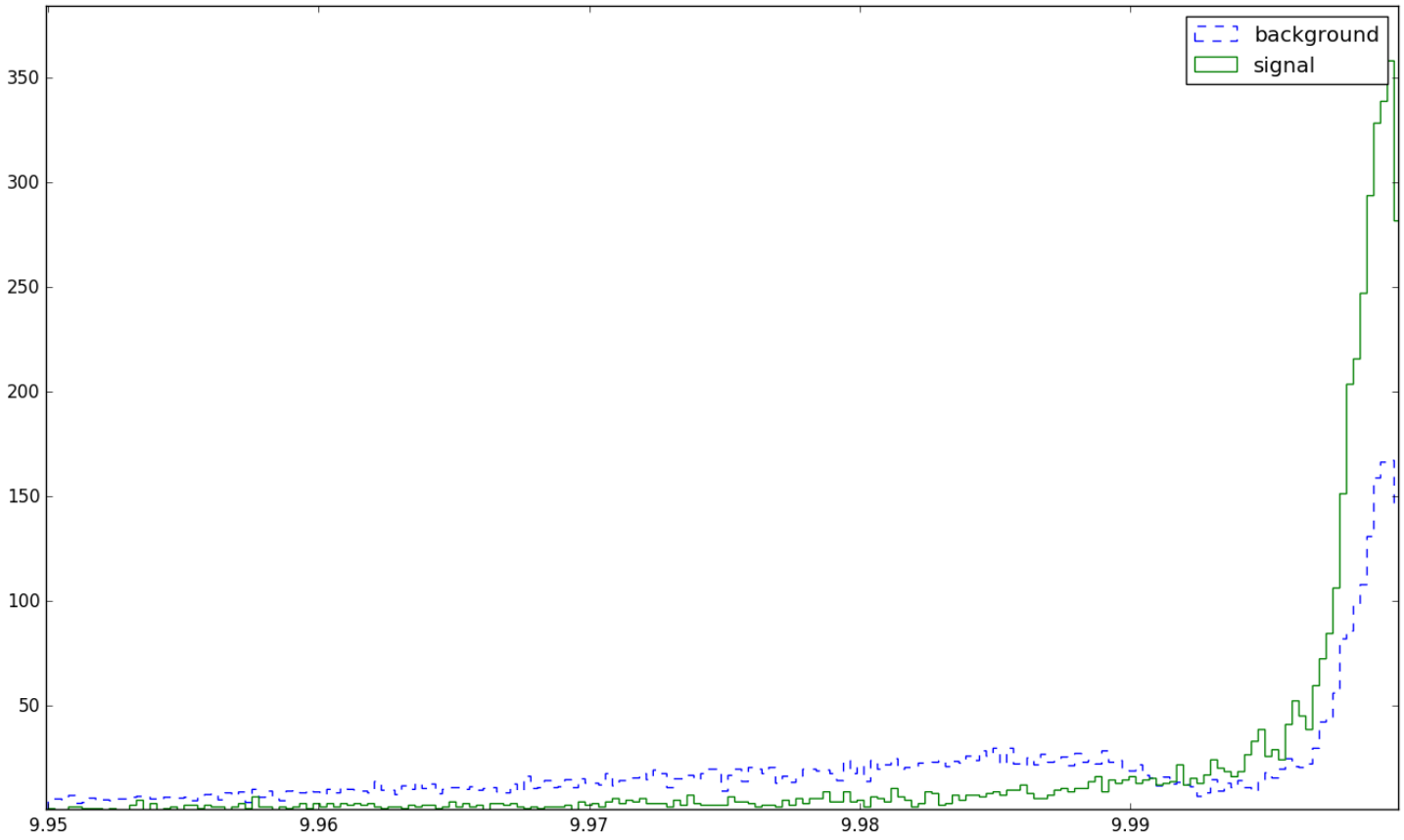
  Tournament selection: select randomly k individuals and keep the best one (here k=3)

- **The process is repeated during 150 generations with 500 individuals**

APPLICATION TO CLAS12 DATA

Discriminating power

$$\left(\cos\left(\phi^{\gamma_1} - \phi^{\gamma_2}\right) + 9\right)\cos\left(\theta^{\gamma_1} - \theta^{\gamma_2}\right)$$



APPLICATION TO CLAS12 DATA

Multiple feature construction

- **Here with vectorial representation, constructing 5 features at once**

$$\|\vec{p_e} + \vec{p_p} + \vec{p_{\gamma_1}}\|$$

$$1 + \cos\left(\angle\left(\vec{p_{\gamma_1}}, \vec{p_e}\right)\right)$$

$$\angle(\vec{p_p} - \vec{p_{\gamma_1}}, \vec{p_{\gamma_1}})$$

$$\frac{\angle(2\vec{p_{\gamma_1}} + 2\vec{p_{\gamma_3}}, -2\vec{p_e} - \vec{p_{\gamma_1}} - \vec{p_{\gamma_2}} + 3\vec{p_{\gamma_3}} + \vec{p_p})}{\angle(\vec{p_e} - \vec{p_{\gamma_1}} + 2\vec{p_{\gamma_2}} - 3\vec{p_{\gamma_3}} - \vec{p_p}, 2\vec{p_e} + 2\vec{p_{\gamma_2}} - 2\vec{p_p})}$$

$$\cos\left(\angle(\vec{p_{\gamma_2}}, \vec{p_{\gamma_1}})\right)$$

Adding all of these 5 features to the initial ones increases the score by 3.4 points