Symbolic regression

Nino Boismenu Nathan Boyer Hubert Gruniaux

January 2025

What is symbolic regression ?

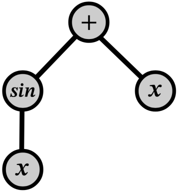


Figure – Expression Tree

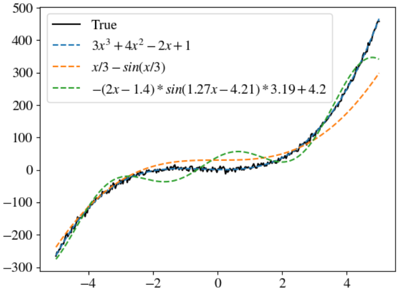


Figure – Regression of $3x^3 + 4x^2 - 2x + 1$ with gaussian noise

The different methods for symbolic regression

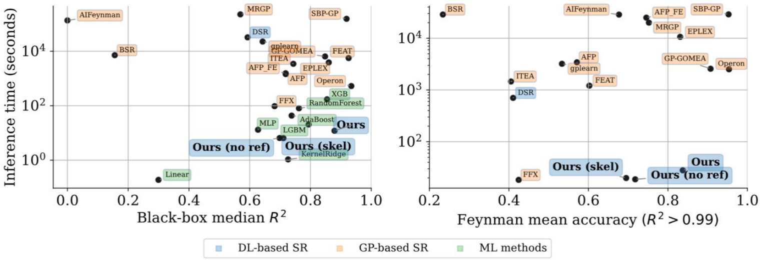
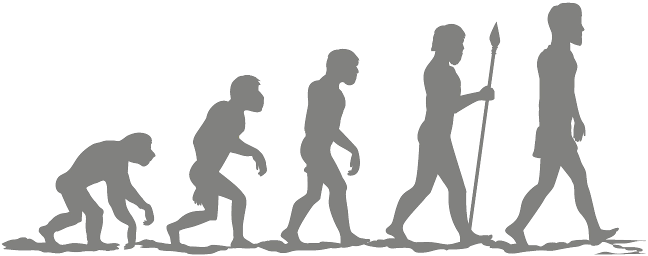
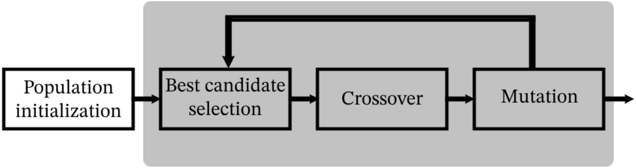


Figure – Different method for symbolic regression. DR : Deep learning SR, GP : genetic programming, ML : Classic machine learning methods. *End-to-end symbolic regression with transformers, P.A Kamienny & al.*

Genetic algorithms





Application to symbolic regression

1: $G \leftarrow$ Random population of N formulas
2: **while** $\min\{l(f, y), f \in G\} > \tau_{\text{target}}$ **do**
3: $C \leftarrow k$ best candidates of G
4: $G_m \leftarrow \lambda_m N$ mutations of C
5: $G_c \leftarrow \lambda_c N$ crossover of C
6: $G_r \leftarrow (1 - \lambda_m - \lambda_c)N$ random formulas
7: $G \leftarrow G_m \bigcup G_c \bigcup G_r$
8: **end while**

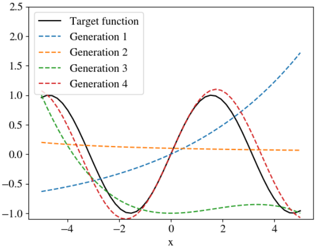
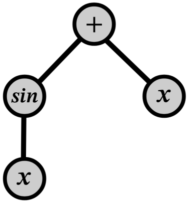


Figure – Evolution of the best candidate across generations.

Representation of mathematical expressions

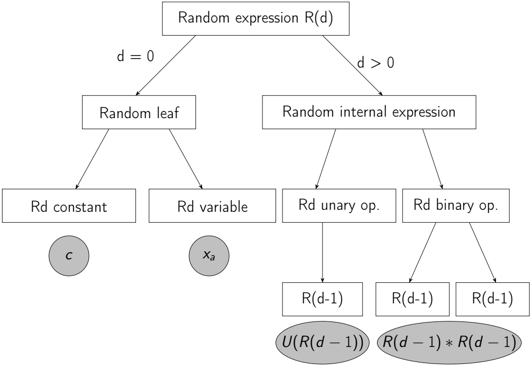
Mathematical expressions as trees, with

- Variables and constants as leaves
- Unary (sin, $\sqrt{\cdot}$, *etc*) and binary ($+$, $\times$, $/$, $-$) operators as nodes




```
class Expression
class ConstantExpression(Expression)
class VariableExpression(Expression)
class BinaryExpression(Expression)
class UnaryExpression(Expression)
class BinaryOp(Enum)
class UnaryOp(Enum)
```

Random generation of formulas



Example of formulas generated by our system :

- $\arctan(e^{1.60-(x*2.59+2.18)} * 4.19 + -1.20) * 1.84 + 0.81$
- $(x * 3.03 + -4.74) * (x * 0.67 + 1.40)$
- $e^{x*-0.17+-0.59} * (-1.51) + -4.84$

The mutations

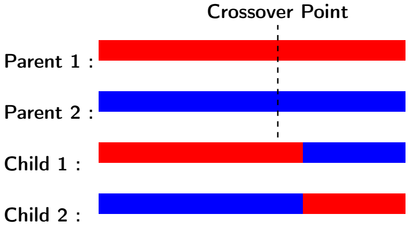
- Random modification to generate a new different formula
- A vast family of mutations implemented in our framework :
  - Operator swapping
  - Operator insertion
  - Operator removing
  - Constant perturbations
  - Variable swapping
  - etc.

The mutations

- Random modification to generate a new different formula
- A vast family of mutations implemented in our framework :
  - Operator swapping
  - Operator insertion
  - Operator removing
  - Constant perturbations
  - Variable swapping
  - etc.
- Smarter mutations :
  - Constant optimization using gradient descent
  - Expression simplification

The cross-overs

- Combine parts of previous formulas to generate a new one
- Only a cross-over strategy implemented in our framework :
  - Pick two random best candidates
  - Pick a random node from one candidate
  - Insert at a random position into the other candidate



The loss

- General loss function based on MSE :

$$l(x, y, f) = \eta \underbrace{\sum_{i=1}^{n} \|f(x_i) - y_i\|_2^2}_{\text{MSE}} + (1-\eta)\overbrace{C(f)}^{\text{Regularization}}$$

with $C(f)$ the *complexity* of f and $\eta \in [0, 1]$ an hyperparameter.

The loss

- General loss function based on MSE :

$$l(x, y, f) = \eta \underbrace{\sum_{i=1}^{n} \|f(x_i) - y_i\|_2^2}_{\text{MSE}} + (1 - \eta) \overbrace{C(f)}^{\text{Regularization}}$$

with $C(f)$ the *complexity* of f and $\eta \in [0, 1]$ an hyperparameter.

- Can extend the loss function :

$$l_{\text{ext}}(x, y, f) = \begin{cases} l(x, y, f) & \text{if } \mathbb{P}(x, y, f) \text{ true,} \\ +\infty & \text{if } \mathbb{P}(x, y, f) \text{ false.} \end{cases}$$

to force multiple variables, non constant formula, etc.

Overfitting and underfitting

- Problem with high-complexity formulas
- Solution, the regularization term in the loss
- Different way to measure *complexity* :
  - Max depth
  - Node count
  - Some arbitrary *smart* function

Results with pure Genetic algorithm : 1D

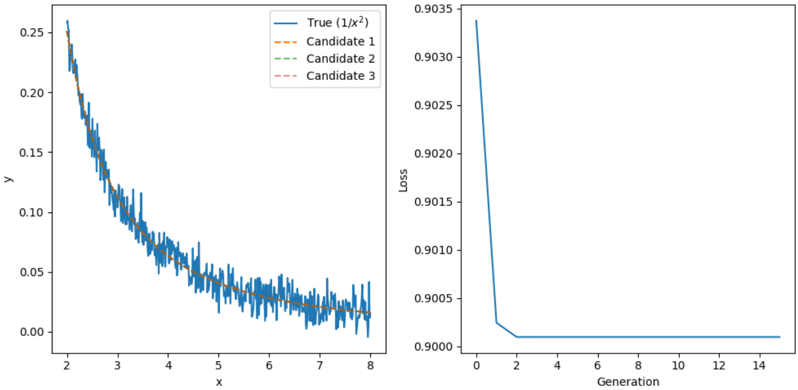


Figure – Symbolic regression for the $1/x^2$ function with 500 samples. The used parameters are a population size of 10000, for 15 iterations.

Results with pure Genetic algorithm : 1D

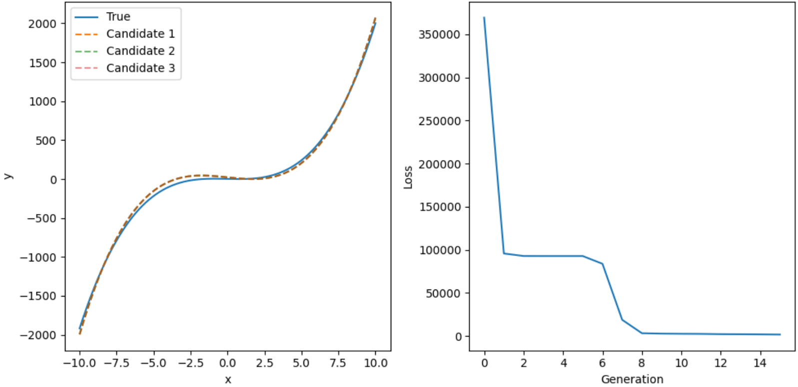
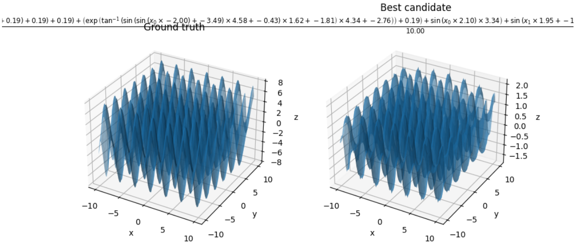
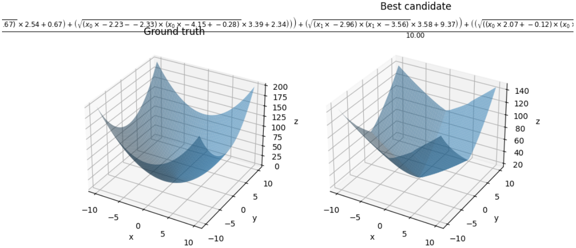


Figure – Symbolic regression for a polynomial function of degree 3 with 500 samples. The used parameters are a population size of 10000, for 15 iterations.

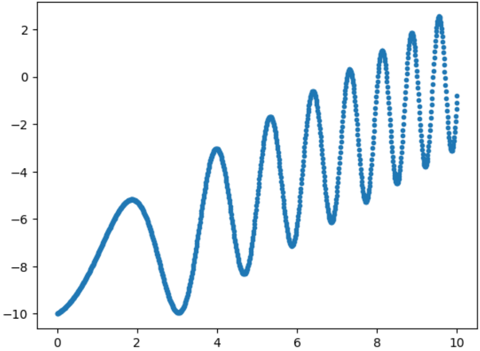
Results with pure Genetic algorithm : 2D





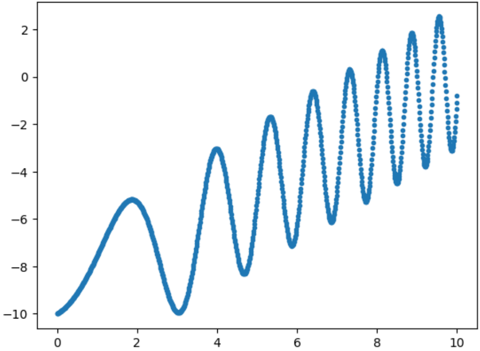
Better guest of initial population

Sometimes, with our human eye, it is easy to see that a function has more chances to include a *cos* than an *exp*.



Better guest of initial population

Sometimes, with our human eye, it is easy to see that a function has more chances to include a *cos* than an *exp*.



Our idea is thus to train an neural network to do the work of the human eye automatically, and detect the presence of certain unary operators in the formula.

Training data

The training data is generated as follows :

1. Generate $30,000$ random formulas of maximum depth 3.

Training data

The training data is generated as follows :

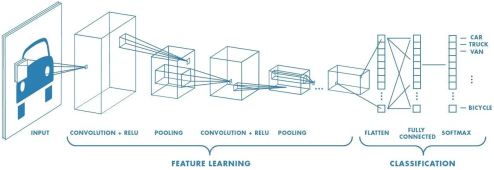
1. Generate $30,000$ random formulas of maximum depth 3.
2. For each formula, evaluate it on values evenly distributed between -10 and 10, this will be the input of the neural network. Data is normalized.

Training data

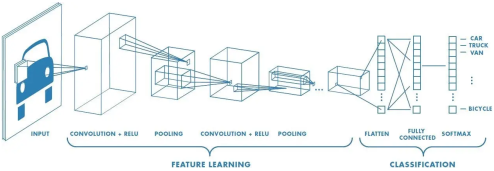
The training data is generated as follows :

1. Generate $30,000$ random formulas of maximum depth 3.
2. For each formula, evaluate it on values evenly distributed between -10 and 10, this will be the input of the neural network. Data is normalized.
3. For each formula, check whether or not it contains each of the following unary operators : $\exp$, $\sin$, $\tan$, $\arcsin$, $\arctan$, $\sqrt{}$, $\log$ This will be the output of the neural network.

Architecture of the CNN



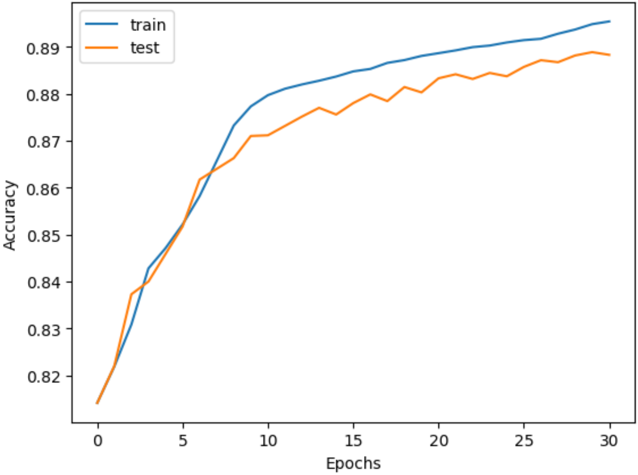
Architecture of the CNN



1. Two 1 dimensional convolutional layers, each followed by a pooling layer
2. 2 linear layers

All layers, except the final one, are followed with a relu activation.

Performance of the CNN



Issues with the CNN

The job of the CNN is complicated because of a number of factors :

Issues with the CNN

The job of the CNN is complicated because of a number of factors :

- Limited computation power causes the use of a rather simple architecture and short training.

Issues with the CNN

The job of the CNN is complicated because of a number of factors :

- Limited computation power causes the use of a rather simple architecture and short training.
- Some unary operators overshadow others in the formula

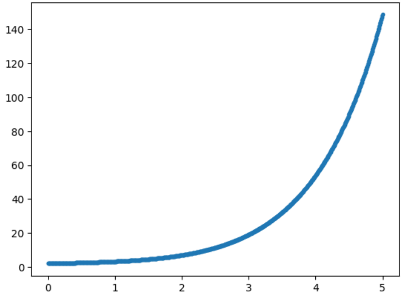
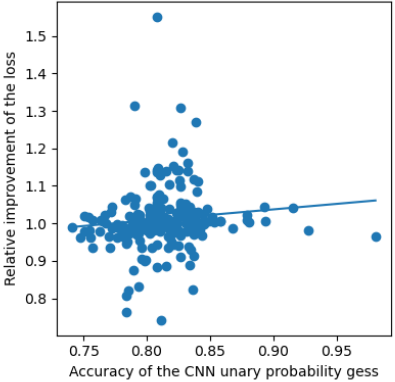


Figure – $\exp x + \cos x$

Does the CNN help ?



We can see there is a correlation between how accurate the CNN is and how well the best function generated using probabilities from the CNN does compared to the best function generated randomly.

Better choice of mutations

We could use the same method to also guide mutations.

- It would increase the impact of a good prediction.
- But also make it very slow for the mutation algorithm to converge if the prediction is bad.

Conclusion

- Satisfying for simple functions
- Support multi-dimensional problems
- Difficulty to choose hyper parameters
- Hybrid version with CNN works but not really faster