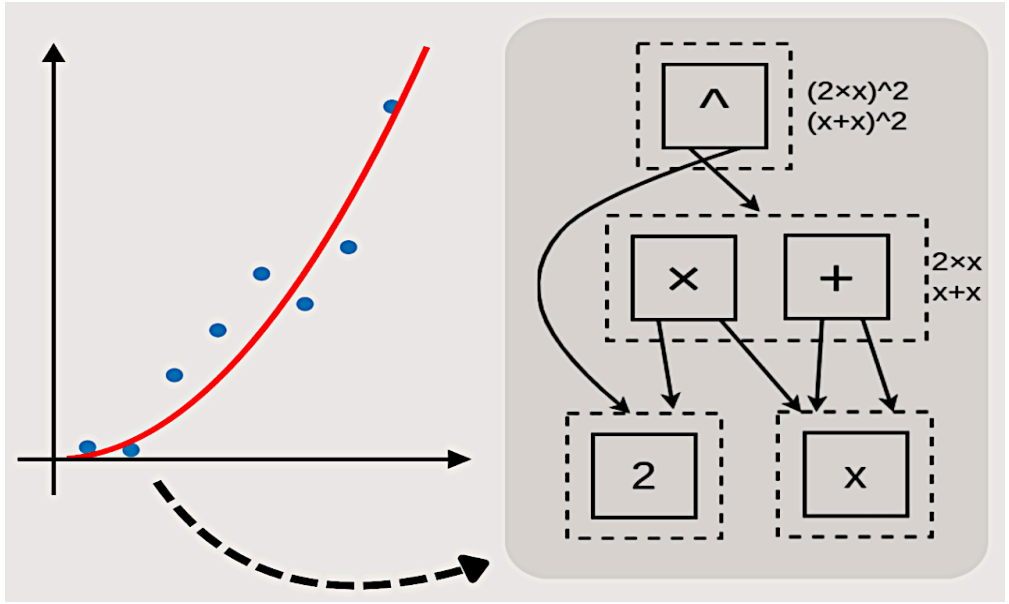
Equality Graphs and Symbolic Regression: A Perfect Match?

F. Olivetti de Franca, Federal University of ABC
G. Kronberger, University of Applied Sciences Upper Austria (FH OÖ)



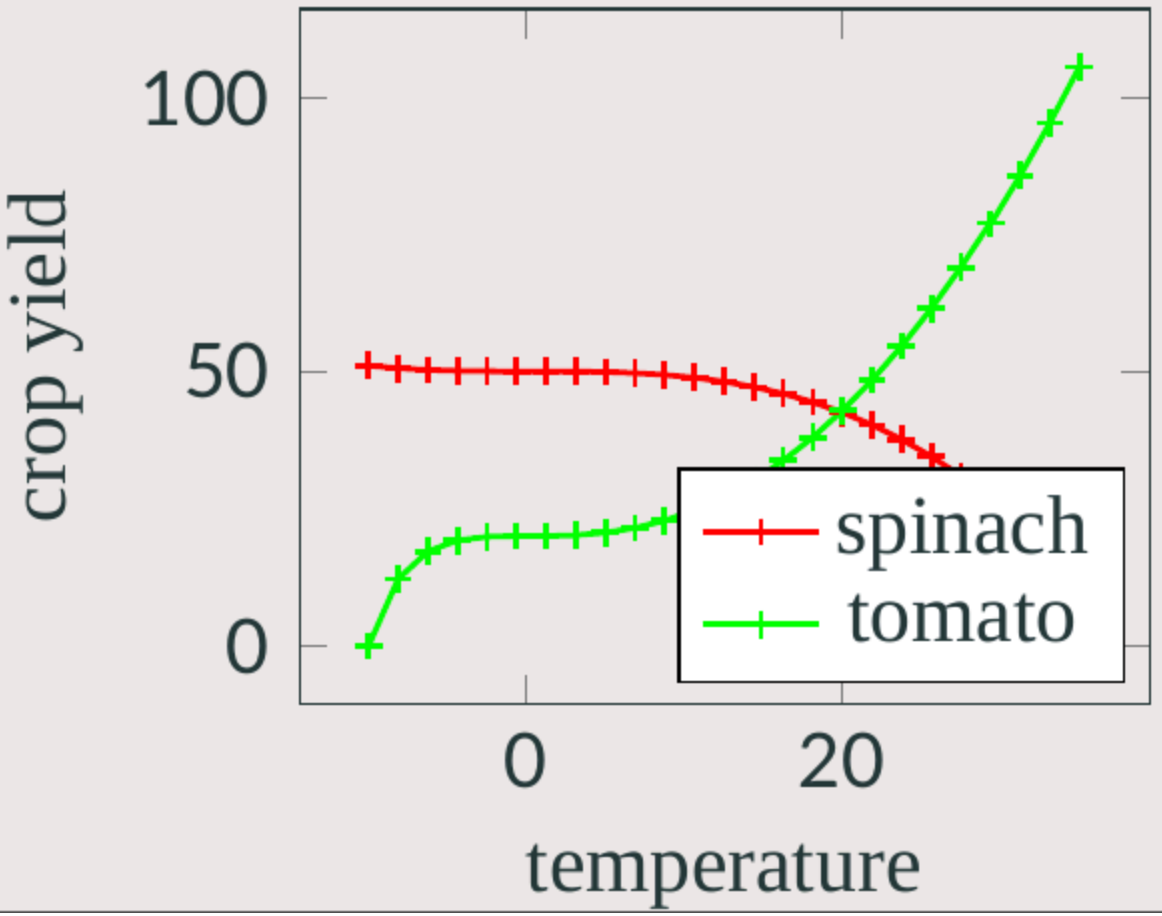
Let's do science!

- Specify a hypothesis.
- Collect data through experiments.
- Describe the measurements with a mathematical function.
- Replicate.

Symbolic Regression

- Search for parametric functions on the form $f(x;\theta)$ describing the observation.

$$f(\mathbf{x},) = \frac{\theta_1 x^3}{\theta_2 + x} + \theta_3$$



Some Challenges

- Easy to generate overparameterized functions
- Search space is not smooth
- There can be multiple equally good solutions

The solution

- e-graphs and equality saturation can help with all these!!

Overparameterization

Fabricio Olivetti de Franca and Gabriel Kronberger. 2023. Reducing Overparameterization of Symbolic Regression Models with Equality Saturation. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23). Association for Computing Machinery, New York, NY, USA, 1064–1072. https://doi.org/10.1145/3583131.3590346

Overparameterized models

Symbolic Regression algorithms are prone to overparametrization:

$$f(\mathbf{x}, \theta) = \theta_1 \exp\left(\theta_2 x_1 + \theta_3\right)$$

- The parameters can assume different values for the same data
- Numerical issues and slow convergence of optimization
- Larger search space for memetic approaches
- Interpretation is hindered

Research Questions

- Is overparametrization really a problem?
- Can EqSat help to reduce overparametrization in SR?
  - How well does it compare with simpler alternatives? (e.g., Sympy)
- Does EqSat always reduce to the optimal number of parameters?
- How fast is EqSat?

Method

- First we run different SR algorithms on two benchmark datasets.
- Then we applied *hegg*, SymPy, and a combination of both in the resulting expressions.
- We compared the ratio of decrease in number of parameters.
- The cost function assigned a value 5 for every parameter and a value of 1 for any other symbol.

Set of Rules

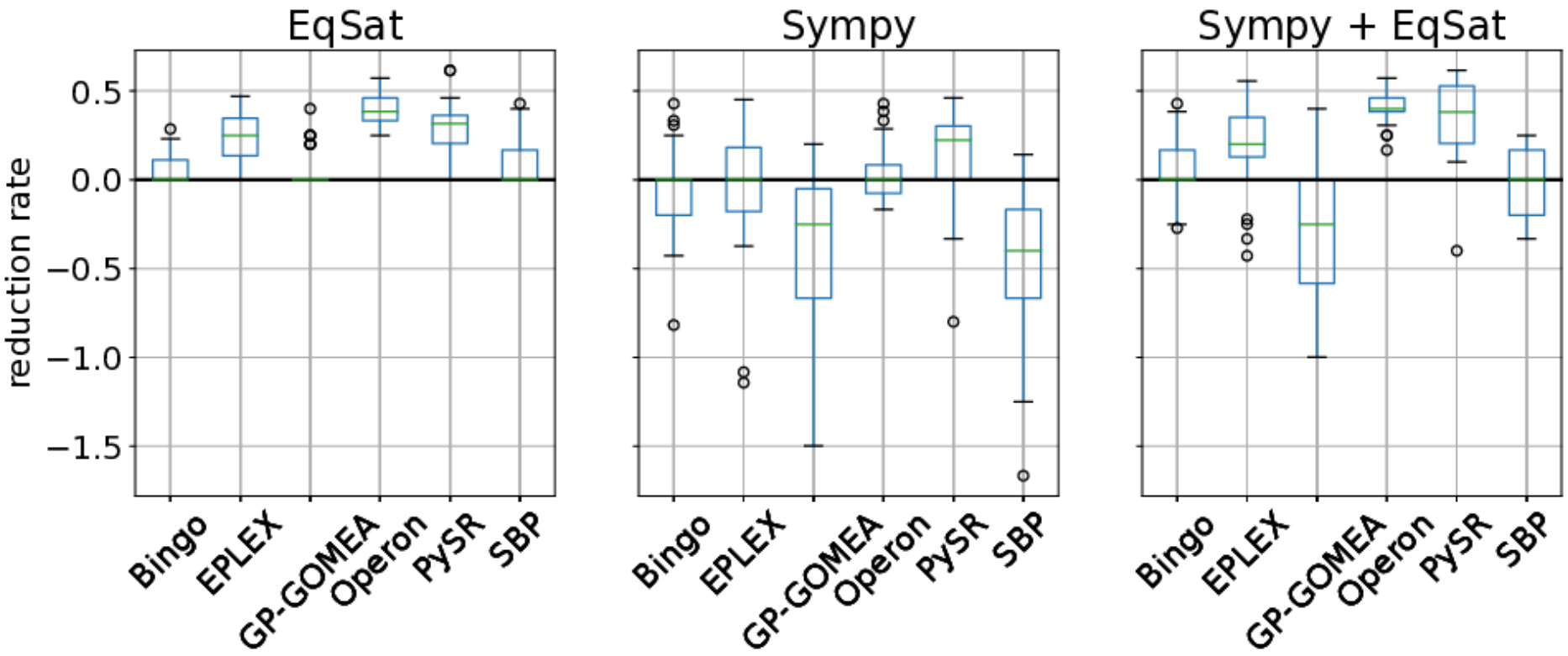
$a + b \to b + a$	$(a * b)/c \to a * (b/c)$
$a * b \to b * a$	$a - 0 \to a$
$a + (b + c) \to (a + b) + c$	$1 * a \to a$
$a * (b * c) \to (a * b) * c$	$0/a \to 0$
$a * (b/c) \to (a * b)/c$	$a - a \to 0$
$0 + a \to a$	$(a * b) + (a * c) \to a * (b + c)$
$0 - a \to -a$	$a - (b + c) \to (a - b) - c$
$0 * a \to 0$	$a - (b - c) \to (a - b) + c$
$-(a + b) \to -a - b$	$(1/a) * (1/b) \to 1/(a * b)$
$a - ((-1) * b) \to a + b$	$a + (-b) \to a - b$
$\log(\exp a) \to a$	

$$a * (1/a) \to 1, \forall a \neq 0$$
$$a/a \to 1, \forall a \neq 0$$
$$\log(a^b) \to b * \log a, \forall a, b > 0$$
$$\log(a * b) \to \log a + \log b, \forall a, b > 0$$
$$\log(a/b) \to \log a - \log b, \forall a, b > 0$$
$$\exp(\log a) \to a, \forall a > 0$$

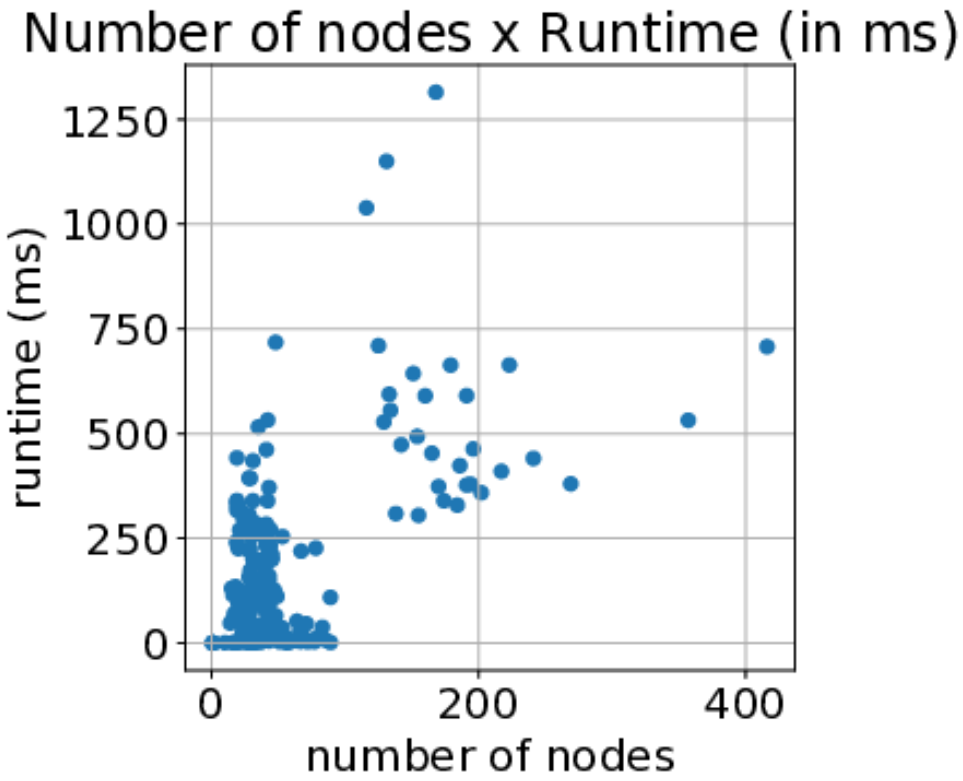
Set of Rules

$$
\begin{aligned}
&(a * x) * (b * y) \to (a * b) * (x * y)\\
&a * x + b \to a * (x + (b/a))\\
&a * x - b \to a * (x - (b/a))\\
&b - a * x \to a * ((b/a) - x)\\
&a * x + b * y \to a * (x + (b/a) * y)\\
&a * x - b * y \to a * (x - (b/a) * y)\\
&a * x + b/y \to a * (x + (b/a)/y)\\
&a * x - b/y \to a * (x - (b/a)/y)\\
&a/(b * x) \to (a/b)/x\\
&x/(b * y) \to (1/b) * x/y\\
&x/a + b \to (x + (b * a))/a\\
&x/a - b \to (x - (b * a))/a\\
&b - x/a \to ((b * a) - x)/a\\
&x/a + b * y \to (x + (b * a) * y)/a\\
&x/a + y/b \to (x + y/(b * a))/a\\
&x/a - b * y \to (x - (b * a) * y)/a\\
&x/a - b/y \to (x - y/(b * a))/a\\
&(b + a * x)/(c + d * y) \to (a/d) * (b/a + x)/(c/d + y)\\
&(b + x)/(c + d * y) \to (1/d) * (b + x)/(c/d + y)\\
&(x - a) \to x + (-a)\\
&(x - (a * y)) \to x + (-a * y)
\end{aligned}
$$

How much can EqSat help?



Is it fast?



Sometimes the set of rules induced some exponential growth of the e-graph...

SRTree: a library for symreg with tree representation

- Haskell library with convenient functions to handle expression trees
- Integrated e-graph and equality saturation, based on hegg and egg

Conveniences

- Convenient fields for the main structure: sorted list of e-classes by fitness, unevaluated e-classes, e-classes that must be refited, height of the best expression of each e-class, information about parameters
- Convenient functions: getter and setter for some of the fields, conversion to an from tree repr., get all possible expressions from one id, get top e-classes within range or given conditions

Avoiding exponential growth

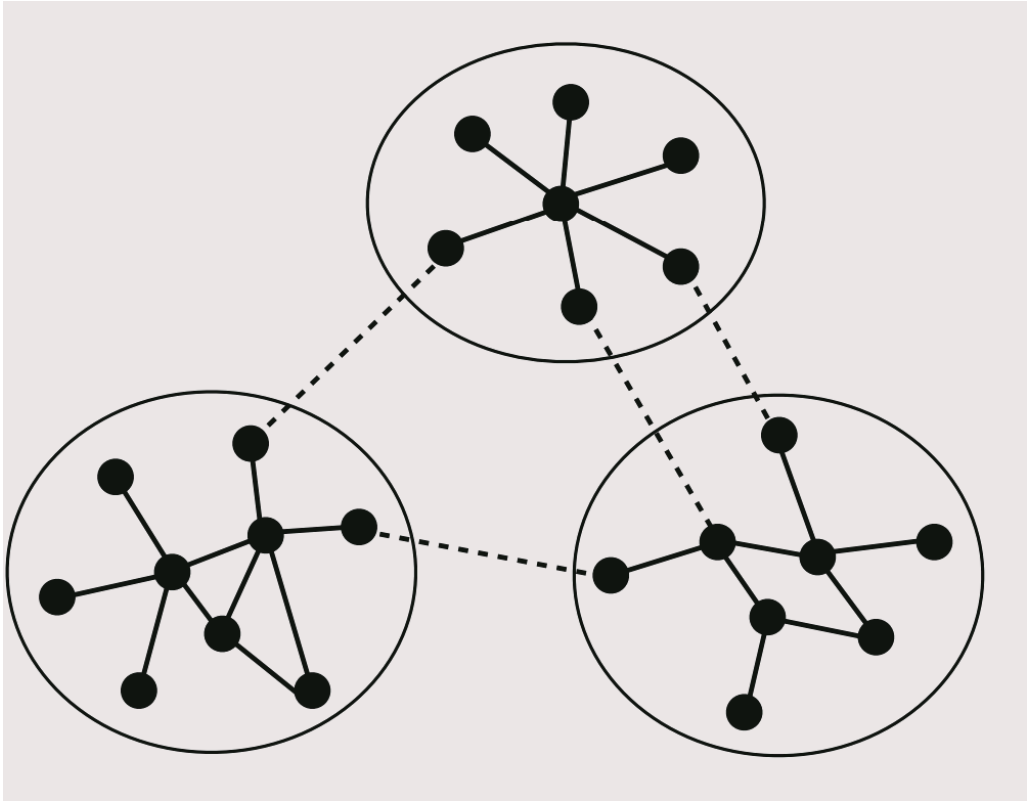
- Stop eq-sat if it reaches an upper limit of e-classes
- Apply the matching rules only if it will merge e-classes (i.e., no new e-class created) (optional)
- Height increase limit (optional)
- This reduced the frequency in which eq-sat induced exponential growth while keeping the final results reasonable for SR.

Navigating the Search space

Fabricio Olivetti de França and Gabriel Kronberger. 2025. Improving Genetic Programming for Symbolic Regression with Equality Graphs. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '25). Association for Computing Machinery, New York, NY, USA, 989–998. https://doi.org/10.1145/3712256.3726383

Neutrality

- **Neutral perturbations:** navigability despite selective pressure.
- **Bloating** introduces new patterns without impacting the fitness.
- New opportunities to find better solutions later.
- $\theta_0 x + e^{\theta_1 + x}$
  $\theta_0 x + e^{\theta_1 + x^{\mathbf{2}}/\mathbf{x}}$



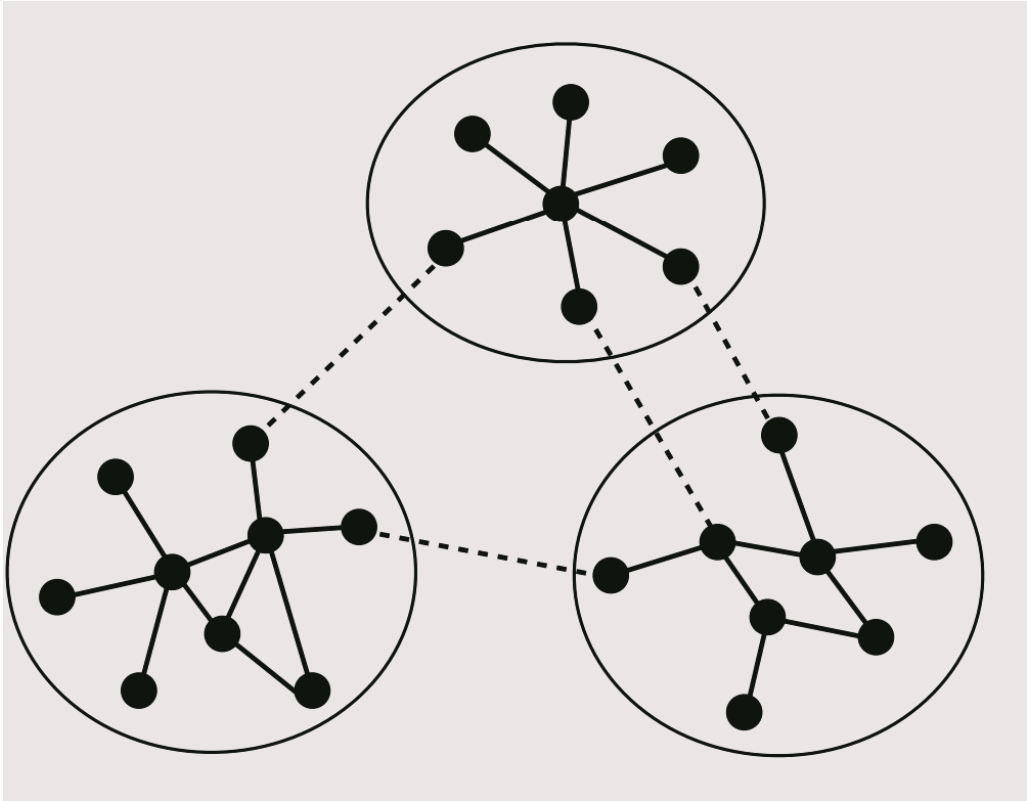
Neutrality

This has other implications:

- waste of evaluation budget
- waste of generations in the same neutral region

Ideally we would be able to:

- detect neutral zones
- **generate** the neutral perturbations
- create a **escape pod** from those regions



EGGP (E-Graph Genetic Programming)

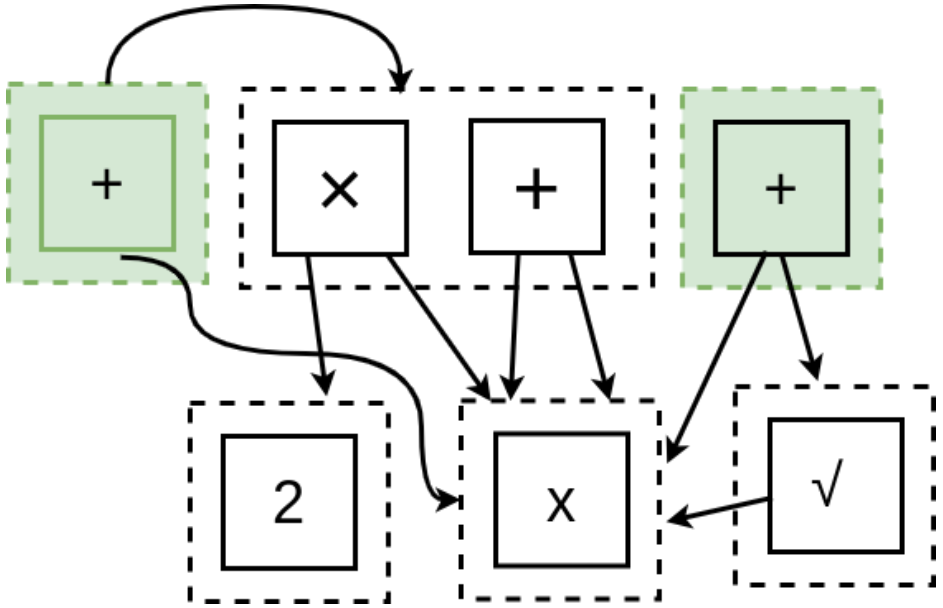
Key differences to GP:

- whole **search history** is kept in memory
- modified crossover and mutation to skip the neutral region

F. Olivetti de Franca, G. Kronberger: Improving Genetic Programming for Symbolic Regression with Equality Graphs, Proc. of GECCO'25, 2025

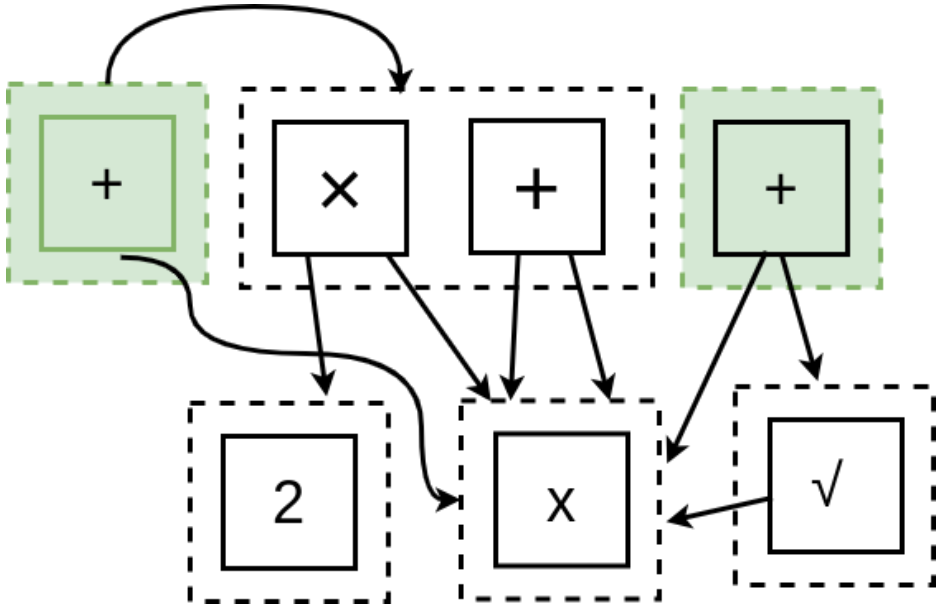
E-Graph Mutation

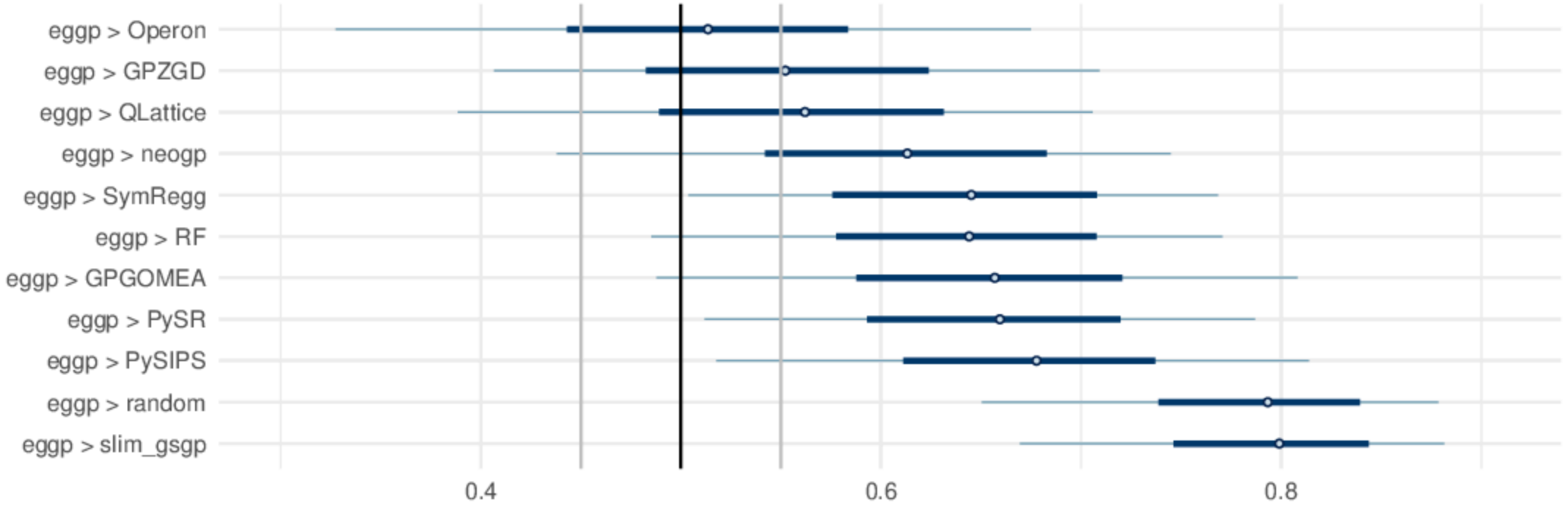
- The green boxes are expressions that were previously evaluated.
- After mutating $x + \sqrt{\mathbf{x}}$ to $x + \mathbf{2x}$, detect revisitation.
- Replace the root $\times$ with any other operator that creates something new ($+, -, \div$).



E-Graph Crossover

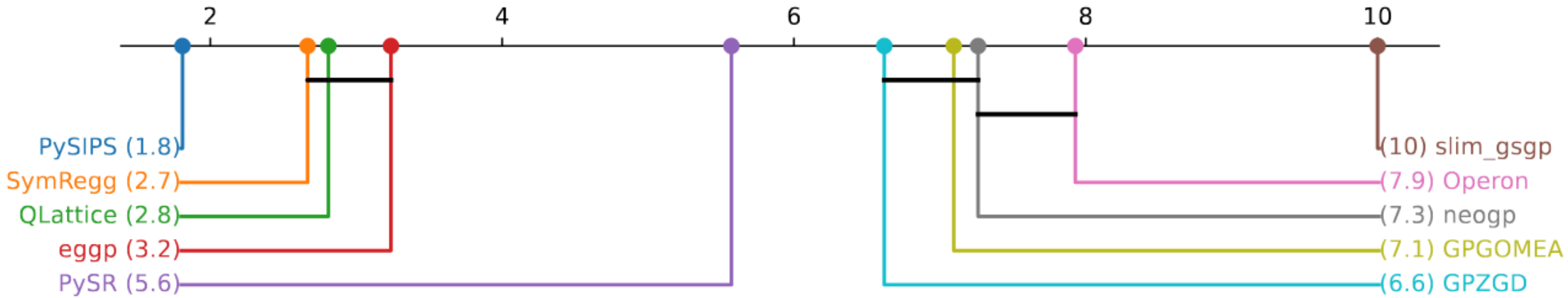
- First parent and crossover point $x + \sqrt{\mathbf{x}}$.
- Second parent $x + 2x$ with the choices of $x + 2x, x, 2x, 2$.
- Replacing the red part with either $x + 2x, x, 2$ will generate something new.





Critical Difference diagram of the average ranks of the tested methods calculated by the average MSE over the 30 runs of each dataset

Results: Model Size



CD diagram of the ranks calculated over average model size.

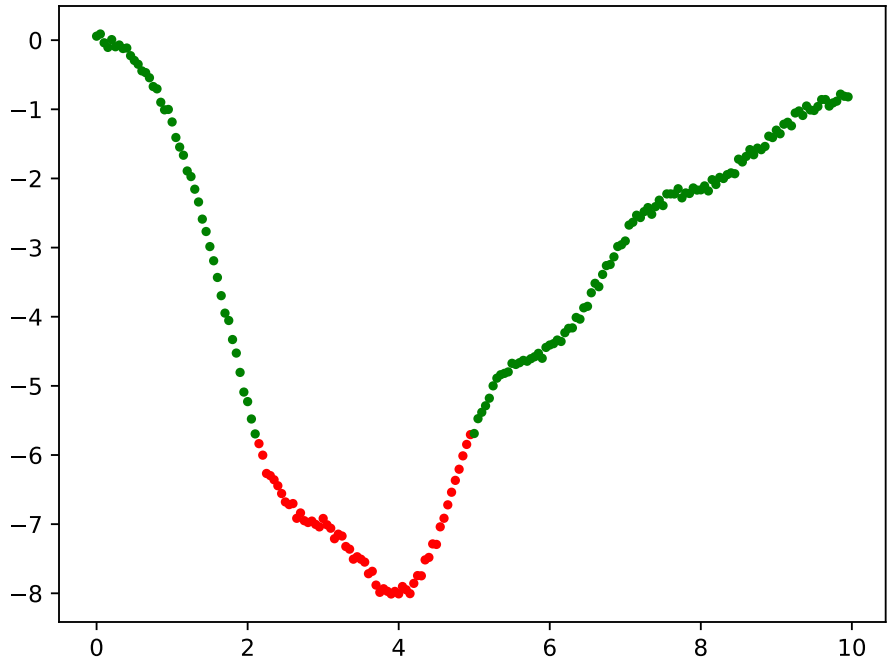
Exploring the probable hypotheses

Fabricio Olivetti de França and Gabriel Kronberger. 2025. REGGression: an Interactive and Agnostic Tool for the Exploration of Symbolic Regression Models. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '25). Association for Computing Machinery, New York, NY, USA, 4–12. https://doi.org/10.1145/3712256.3726385

r🥚ression: Playground for e-graphs and equality saturation in SR

- Database storing multiple models using **e-graphs**.
- Pattern matching ability.
- Building blocks explorations.
- Modularity detection.
- Import expressions from:
  - Operon
  - PySR
  - GOMEA
  - etc.
- Pandas DataFrame.

More info at **https://github.com/folivetti/reggression**



Let's fit!

- Nonlinear relationship.
- The training set is insufficient to guarantee an unique global optima.

In any case, the purpose here is to show how we can use r🥚ression to explore alternative models.

Laying the egg 🥚

We can create an initial e-graph for this dataset using `eggp`:

```
from eggp import EGGP
import pandas as pd

reg = EGGP(gen=200, nPop=200, maxSize=25, \
      nonterminals="add,sub,mul,div,log,power,sin,cos,abs,sqrt", \
      simplify=True, optRepeat=2, optIter=20, folds=2,  \
      dumpTo="vlad.egg")
reg.fit(x_sel, y_sel)
```

Some observations

- The non-terminal set is large to generate many different alternative models.
- We could possibly find better models with proper settings.
- The maximum size is larger than the true equation.

We are saving the final e-graph into the file named `vlad.egg` so we can explore it after the search.

Hatching the egg 🐣

Now, let's load the e-graph into `r🥚ression`:

```
from reggression import Reggression
egg = Reggression(dataset="vlad.csv", loadFrom="vlad.egg")
```

Top-N models, the third one will surprise you!

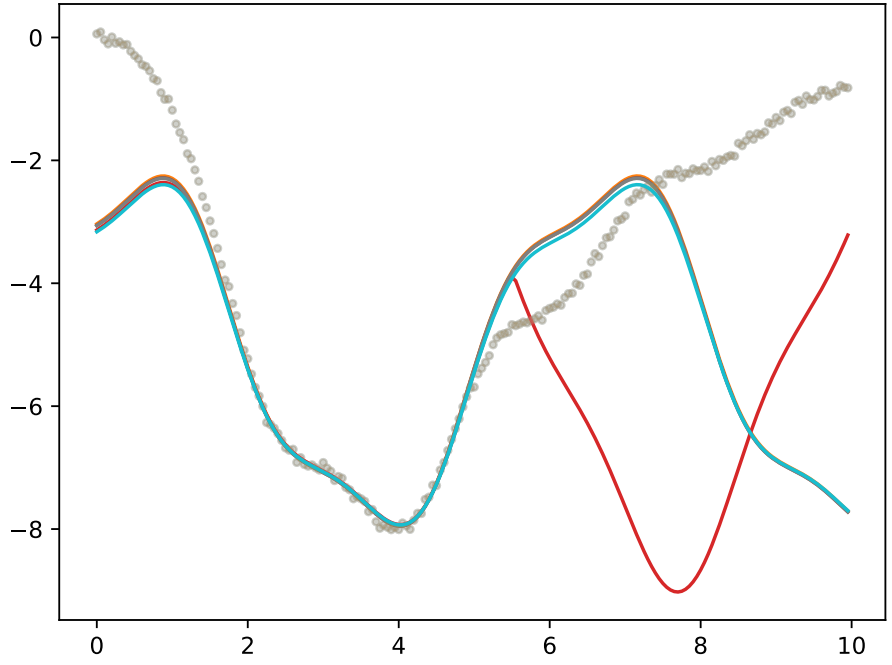
If we look at the top-5 models, we can see small variations of the top performing with similar fitness (negative MSE) values.

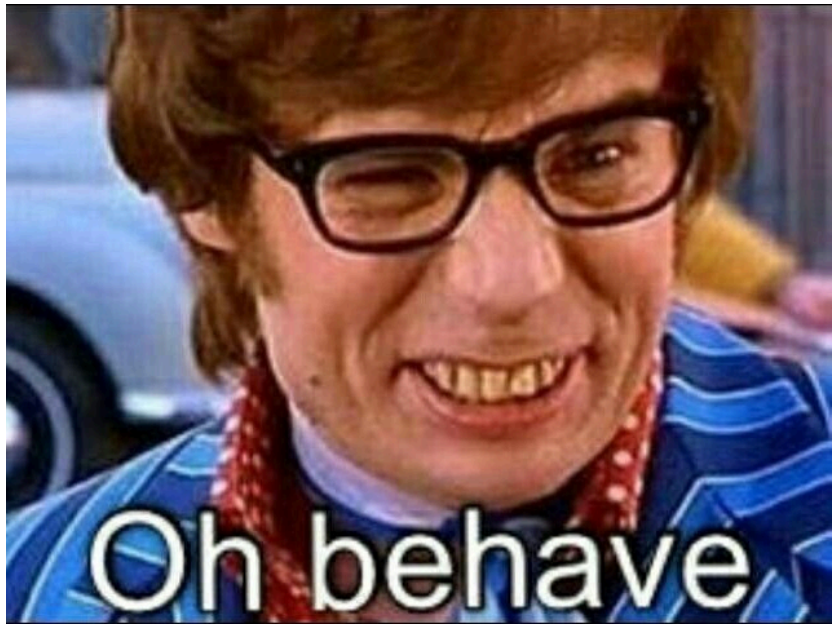
```
egg.top(5)[["Latex", "Fitness", "Size"]]
```

Latex	Fitness	Size
$((\cos(\cos(x)) \cdot (\theta_0 \cdot \cos(((\theta_1 - (x + \theta_2)) + \theta_3)))) + \theta_4)$	-0.00415306	16
$(\theta_0 + (\cos(\cos(x)) \cdot (\cos((\|\| ((x + \theta_1) + \theta_2) \|\| +\theta_3)) \cdot \theta_4)))$	-0.00425244	18
$(((\cos(\cos(x)) \cdot (\cos(((\theta_0 - (x + \theta_1)) + \theta_2)) \cdot \theta_3)) + \theta_4) + \theta_5)$	-0.00430326	18
$\left(\theta_0 + \left(\left(\cos(\cos(x)) \cdot \left(\cos(\left(\| \left(\sqrt{x}^2 + \theta_1\right) \| +\theta_2\right)) \cdot \theta_3\right)\right) + \theta_4\right)\right)$	-0.00430774	19
$(\theta_0 + (\cos(\cos(x)) \cdot (\theta_1 \cdot \cos(((\theta_2 - x) + \theta_3)))))$	-0.0043503	14

Behave!

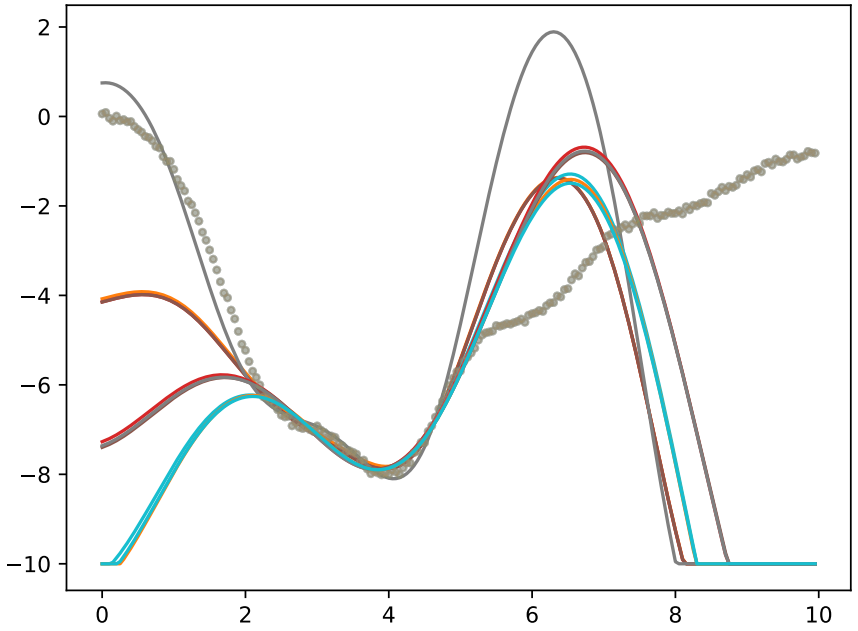
Some of these functions behave similarly while others display a different behavior when looking outside of the training region:





Filter and plot!

We can retrieve top expressions filtering by size:




```
model_top(egg.top(n=10, filters=["size <= 10"]), n, x, y)
```

Rich Gets Richer

Let us investigate the distribution of tokens from the top 1000 generated expressions.:

```
egg.distributionOfTokens(top=1000)
```

Pattern	Count	AvgFit	Pattern	Count	AvgFit	Pattern	Count	AvgFit
x0	2604	-0.00359749	t6	1	-0.013187	Exp(v0)	45	-0.0118458
t0	1006	-0.009312	Abs(v0)	465	-0.00810496	Cube(v0)	38	-0.00867039
t1	981	-0.00941213	Sin(v0)	74	-0.0115615	(v0 + v1)	3405	-0.00275121
t2	955	-0.00937039	Cos(v0)	3029	-0.00309273	(v0 - v1)	351	-0.00848634
t3	806	-0.00893546	Sqrt(v0)	32	-0.00845579	(v0 * v1)	2139	-0.0042415
t4	466	-0.00910986	Square(v0)	27	-0.00967352	(v0 / v1)	68	-0.00815694
t5	144	-0.00786632	Log(v0)	10	-0.00972384			

Is tough being exponential!

Sine and cosine are ranked next, while the exponential is rarely used and particularly with a worse average fitness than the other tokens.

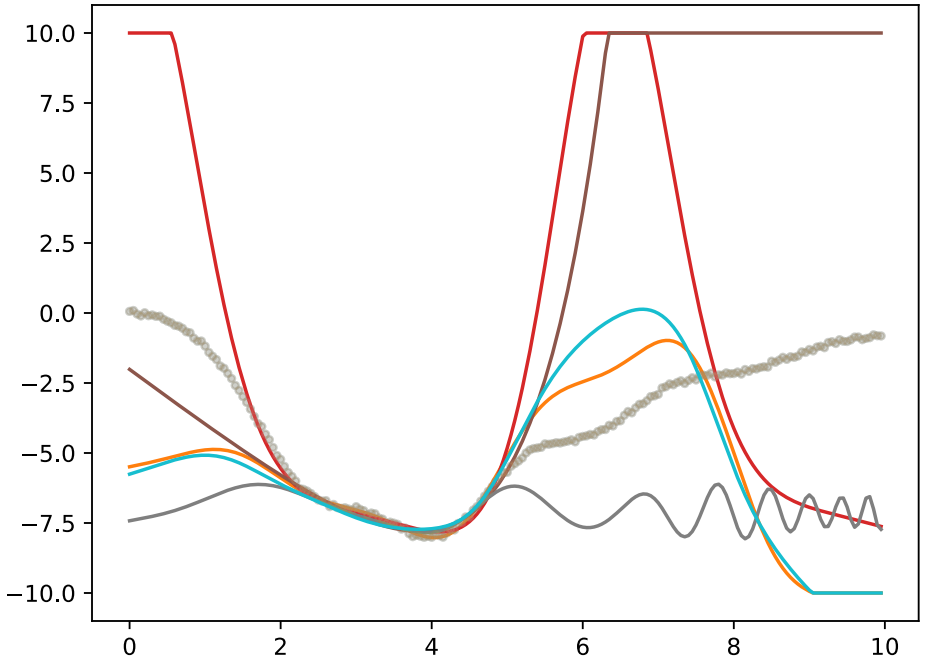
The reason for this could be that fitting parameters inside an exponential function can be tricky depending on the initial values.

Let's check it out!

We can verify that by plotting the top 5 expressions with the pattern $e^{\square_0}\square_1$ with the command:

```
egg.top(n=n, pattern="exp(v0)*v1")
```

and as we can see, still not a very good fit, as expected.



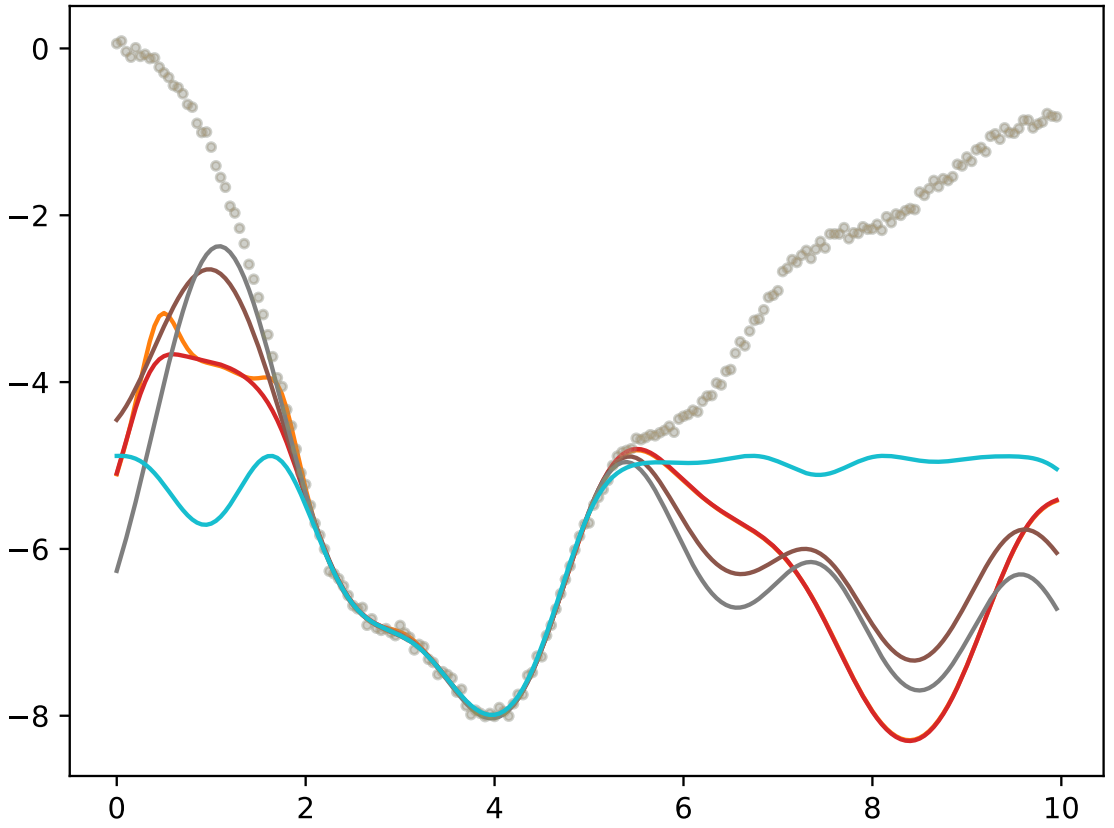
With a little help from my friends 🐣🐤

We can try our luck with another SR method, such as Operon [4], and insert the obtained expressions into the e-graph:

```
from pyoperon.sklearn import SymbolicRegressor
regOp = SymbolicRegressor(objectives=['mse','length'], max_length=20, \
       allowed_symbols='add,sub,mul,div,square,sin,cos,exp,log,sqrt,abs,constant,variable')
regOp.fit(x_sel, y_sel)
f = open("equations.operon", "w")
for eq in regOp.pareto_front_:
  eqstr = regOp.get_model_string(eq['tree'])
  fitness = -eq['mean_squared_error']
  print(f"{eqstr},{fitness},{fitness}", file=f)
f.close()
egg.importFromCSV("equations.operon")
```

Top-5 with Operon

Still no luck! But we didn't make things easy for SR anyway!



Let's cheat!

We can insert the ground-truth expression to see whether the parameter optimization is capable of converging to the true parameters and if the fitness is better than what we have.

```
egg.insert("exp(x0/t0)*(x0^3)*(cos(x0)*(sin(x0)^2)-t1)")
```

Latex	Fitness	Parameters
$\left(\left(x^{3.0} \cdot \left(\left(\cos\left(x\right) \cdot \sin\left(x\right)^{2.0}\right) + \theta_{0}\right)\right) \cdot e^{\left(x \cdot \theta_{1}\right)}\right)$	-0.00256414	[-3.15, -0.83]

It's all the same, no matter where you are 🐥🐥🐥

We can also use r🥚ression to check whether two or more expressions are equivalent. Let's say we want to see whether $(x+3)^2 - 9$ and $x(x+6)$ are the same.

First, we create an empty e-graph:

```
newegg = Reggression(dataset="vlad.csv", loss="MSE")
```

They are different after all

Next, we add both expressions while storing their e-class ids:

```
eid1 = egg.insert("(x0 + 3)**2 - 9").Id.values[0]
eid2 = egg.insert("x0*(x0 + 6)").Id.values[0]
print(eid1, eid2)
> 6, 9
```

Initially, their ids are going to be different, since until now they are distinct to each other as far as the e-graph is concerned.

Or maybe not...

Now, the main idea is that we run equality saturation to produce all the equivalent forms of each one of these expressions following a set of rules, such as:

$$(x + y)^2 \rightarrow x^2 + y^2 + 2xy$$

> If the set of rules are sufficient to produce at least one common expression departing from the first and from the second expressions, they will eventually be merged, and their e-class id will become the same.

Equality Saturation for You!

We can run some iterations of equality saturation using the command:

```
egg.eqsat(5)
```

And, now, their ids should be the same!

```
print("Id of the first equation: \n", egg.report(eid1).loc[0:1, ["Info", "Training"]])
print("Id of the second equation: \n", egg.report(eid2).loc[0:1, ["Info", "Training"]])
> Id of the first equation: 16
> Id of the second equation: 16
```

Multiverse

After running equality saturation, we can also retrieve a sample of the equivalent expressions for that e-class id:

```
egg.getNExpressions(eid1, 10)
```

Leading to:

$((6.0 + x) * x)$	$((x + 6.0) * x)$
$((x * 6.0) + (x^2))$	$((x * 6.0) + (x^2))$
$(0.0 + ((6.0 + x) * x))$	$(0.0 + ((x + 6.0) * x))$
$((2.0 * (x * 3.0)) + (x^2))$	$((2.0 * (3.0 * x)) + (x^2))$
$(((x * 3.0) * 2.0) + (x^2))$	$(((3.0 * x) * 2.0) + (x^2))$

Structural Similarity

We can also measure the similarity between two expressions by measuring the percentage of shared e-classes ids.

```
[17…] from reggression import Reggression
      import pandas as pd

      pd.set_option('display.max_colwidth', 100)
      df = pd.read_csv("datasets/nikuradse_1.csv")
      egg = Reggression(dataset="datasets/nikuradse_1.csv", loss="MSE")

      Calculating DL...
      Welcome to r🥚ression

[31…] id1 = egg.insert("x0+x1")['Id'].values[0]

[32…] id2 = egg.insert("(x1+x0)*x1")['Id'].values[0]

[20…] egg.eqsat(2)

[20…] —

[35…] ecs1 = egg.subtrees(id1)['Id'].values

[36…] ecs2 = egg.subtrees(id2)['Id'].values

[37…] ecs1, ecs2

[37…] (array([3, 1, 0]), array([4, 3, 1, 0]))
```

Code and Resources

Python library and CLI

> pip install eggp
> pip install reggression
> pip install symregg

https://github.com/folivetti/eggp
https://github.com/folivetti/reggression
https://github.com/folivetti/symregg