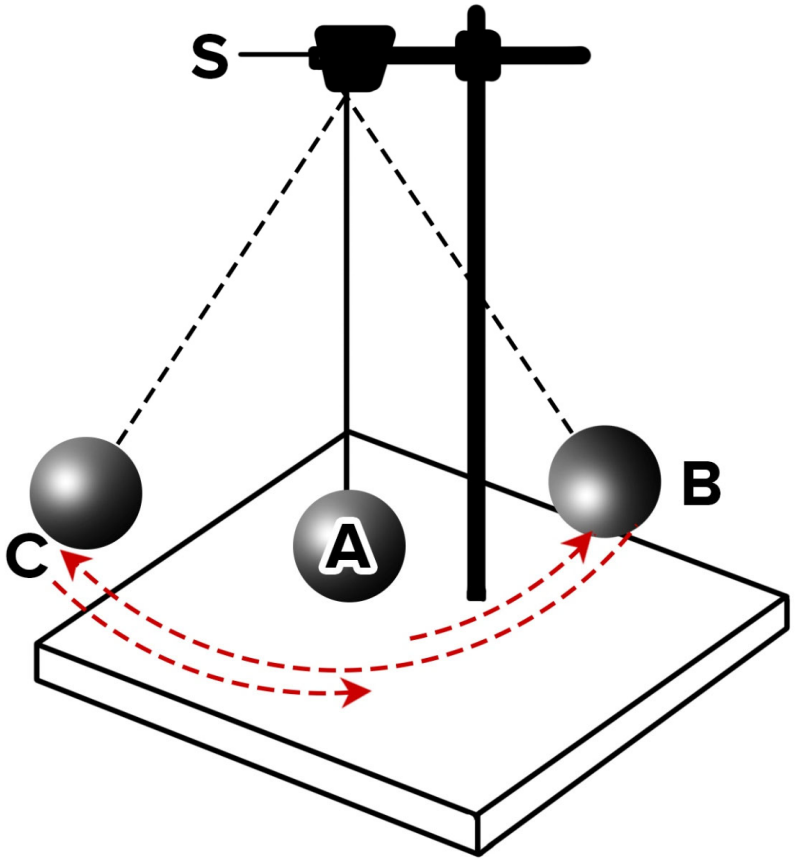
THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 10: Symbolic Regression and Physics Discovery

Sergei V. Kalinin

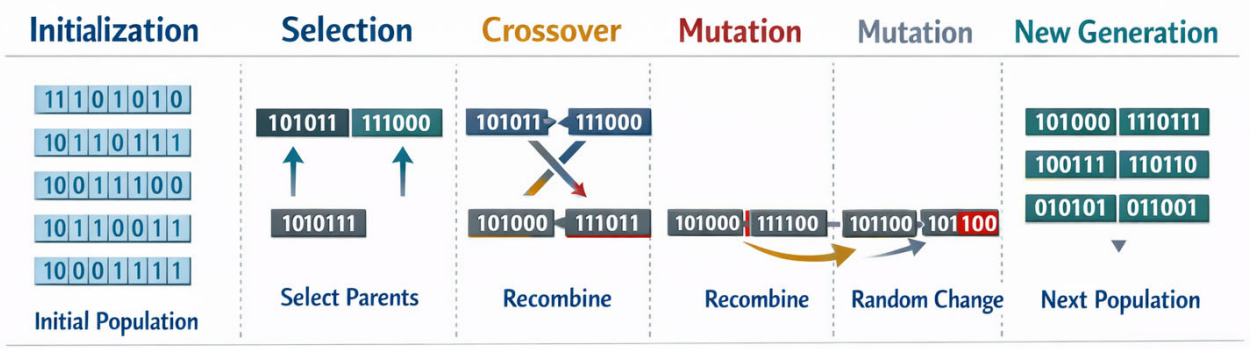
Back to functions from data

- If the function is known, we can fit it to data
- If we have several possible functions, we can fit all of them and compare the quality of fits
- We can also make some judgements based on the parameter values
- But what if we do not know the functions?



Review from last class: Genetic Algorithm (GA)

- Many candidates
- **Selection:** Some works better → something good about these candidates → let's generate new solutions based on them
- Blend the features (genes) of good candidates (crossover) + add some surprise (mutation)
- Evaluate the newly generated in the next cycle



Review from last class: why GA?

Core Advantages

- Works without gradients (black-box optimization)
- Handles discrete, categorical, and mixed variables
- Performs global search instead of local descent
- Robust to noisy objective functions
- Flexible representation (bitstrings, vectors, trees, programs)

Use GA when your problem has:

- Non-differentiable objective functions
- Discrete search spaces
- Combinatorial structure
- Many local optima
- Unknown model structure

Typical Problem Types Ideal for GA

- Architecture search
- Controller tuning
- Hyperparameter optimization
- Experimental design
- Symbolic regression
- Materials discovery search spaces

https://www.human-competitive.org/awards

Question to answer with symbolic regression (SR)

We have some observations x and y. How to find the best analytic function (symbolic representation) describes their relation?

Step 1: define the question:

Discover symbolic (analytical) expression of

$$y = f(x)$$

Based on data of (x, y).

Step 2: Choose the building blocks (search space)

You define what expressions are allowed:

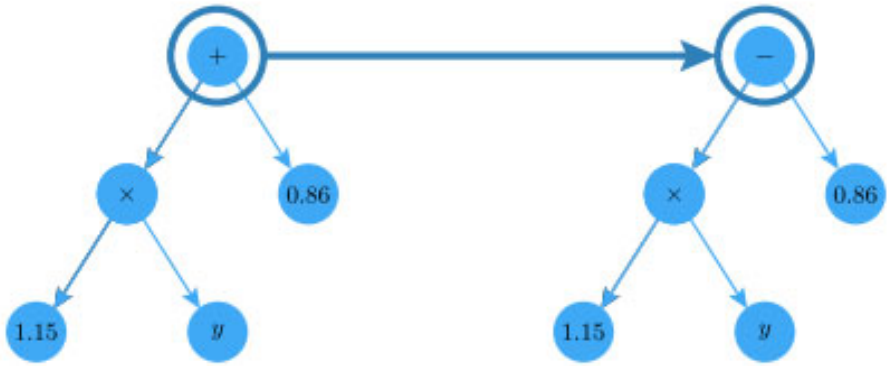
- **Variables**: $x_1, x_2, \ldots$
- **Constants**: numerical values (either fixed or optimized)
- **Operators**:
  - Binary: $+, -, \times, \div$
  - Unary: sin, cos, exp, log, $\sqrt{}$, square, etc.

This defines the "language" of equations SR is allowed to speak → the genes

https://arxiv.org/pdf/2305.01582.pdf

Step 3: Choose a representation (usually expression trees)

An equation is represented as a **tree**:
This makes it easy to modify equations by changing subtrees (gene mutation and crossover)



$1.15y + 0.86$ $1.15y - 0.86$

Step 4: Initialize a population of random equations.

- small trees initially (to avoid huge nonsense)
- randomly chosen operators/variables/constants → try to be diverse and representative

Think: a "pool of hypotheses."

https://arxiv.org/pdf/2305.01582.pdf

Step 5: Evaluate each equation (fitness)

For each candidate function f_i, you compute the fitness based on:

1. The loss: MSE $= \frac{1}{N}\Sigma(\widehat{y_i} - y)^2$, where y is the measurement, and $\widehat{y_i}$ is the prediction of $f_i(x)$
2. The complexity:
   - number of nodes in the tree
   - weighted operator cost

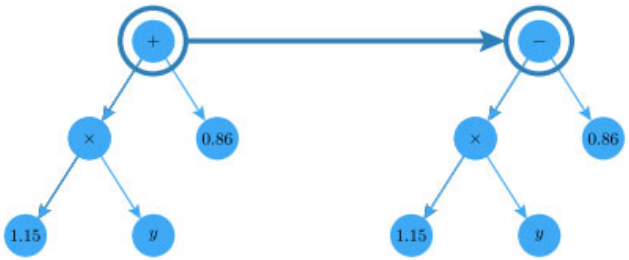
Single-objective fitness: fitness=error+λ·complexity

Or you can build a pareto front to balance error and complexity

https://arxiv.org/pdf/2305.01582.pdf

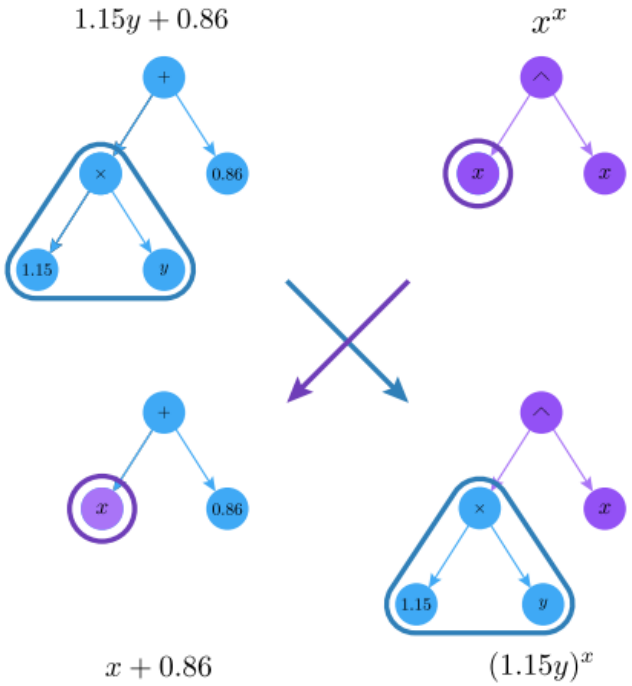
Step 6: Create new equations by genetic operators

Mutation



$1.15y + 0.86$ $1.15y - 0.86$

Crossover



$x + 0.86$ $(1.15y)^x$

Crossover (subtree swap):

- Take two parent trees and swap random subtrees → child equation.

Mutation (random edits)

- replace an operator
- replace a subtree
- insert a new subtree
- tweak a constant
- change variable x1→x2

https://arxiv.org/pdf/2305.01582.pdf

Step 7: (optional) Optimize constants inside each structure

Many SR methods refine constants after proposing a structure.

Example:

$$y = \mathrm{a} \cdot \sin(b \cdot x) + c$$

Once the structure is chosen, find a,b,c that minimize error (using local optimization / regression).

This is why SR can be both:

- structural search (global)
- parameter fitting (local)

Step 8: iteration and output

Output: a set of candidate equations (not just one)

- simplest acceptable model
- most accurate model
- best generalization model

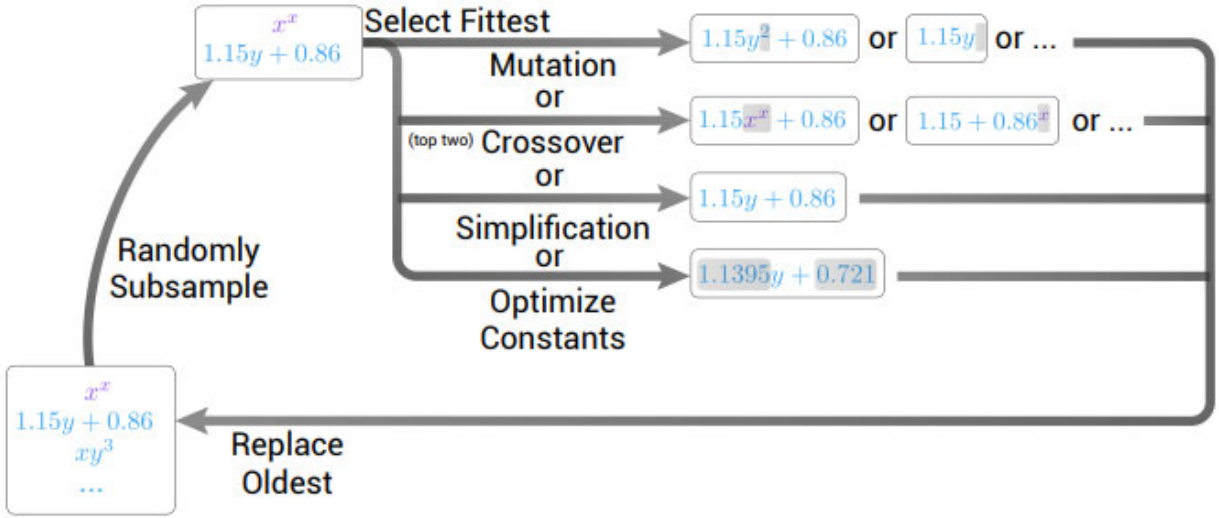
This is typically shown as a Pareto frontier:

x-axis: complexity

y-axis: error

https://arxiv.org/pdf/2305.01582.pdf

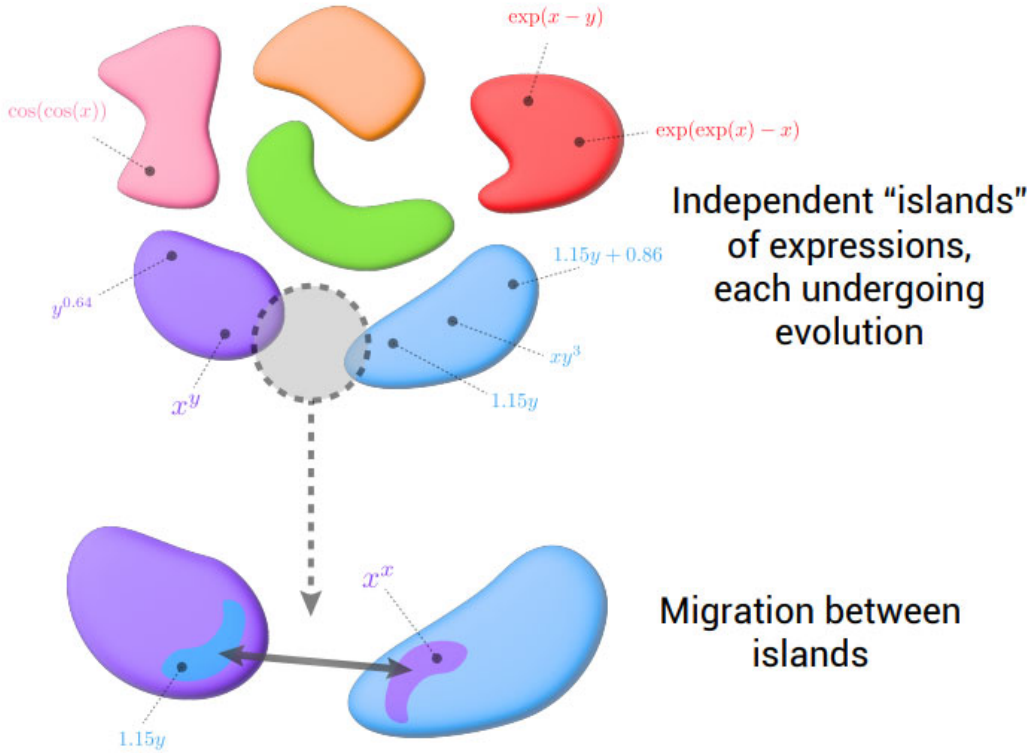
PySR



The inner loop of PySR. A population of expressions is randomly subsampled. Among this subsample, a tournament is performed, and the winner is selected for breeding: either by mutation, crossover, simplification, or explicit optimization. Examples of mutation and crossover operations are visualized in figs. 1 and 2.

https://arxiv.org/pdf/2305.01582.pdf

PySR



The outer loop of PySR. Several populations evolve independently according to the algorithm described in fig. 3. At the end of a specified number of rounds of evolution, migration between islands is performed.

https://arxiv.org/pdf/2305.01582.pdf

PySR Benchmarking

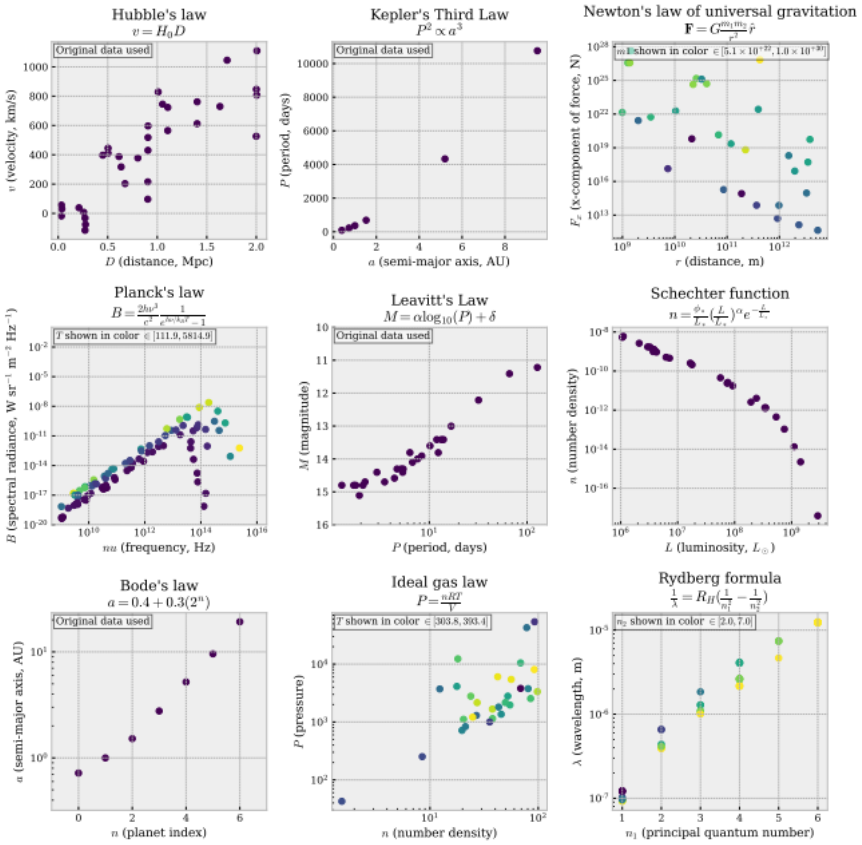
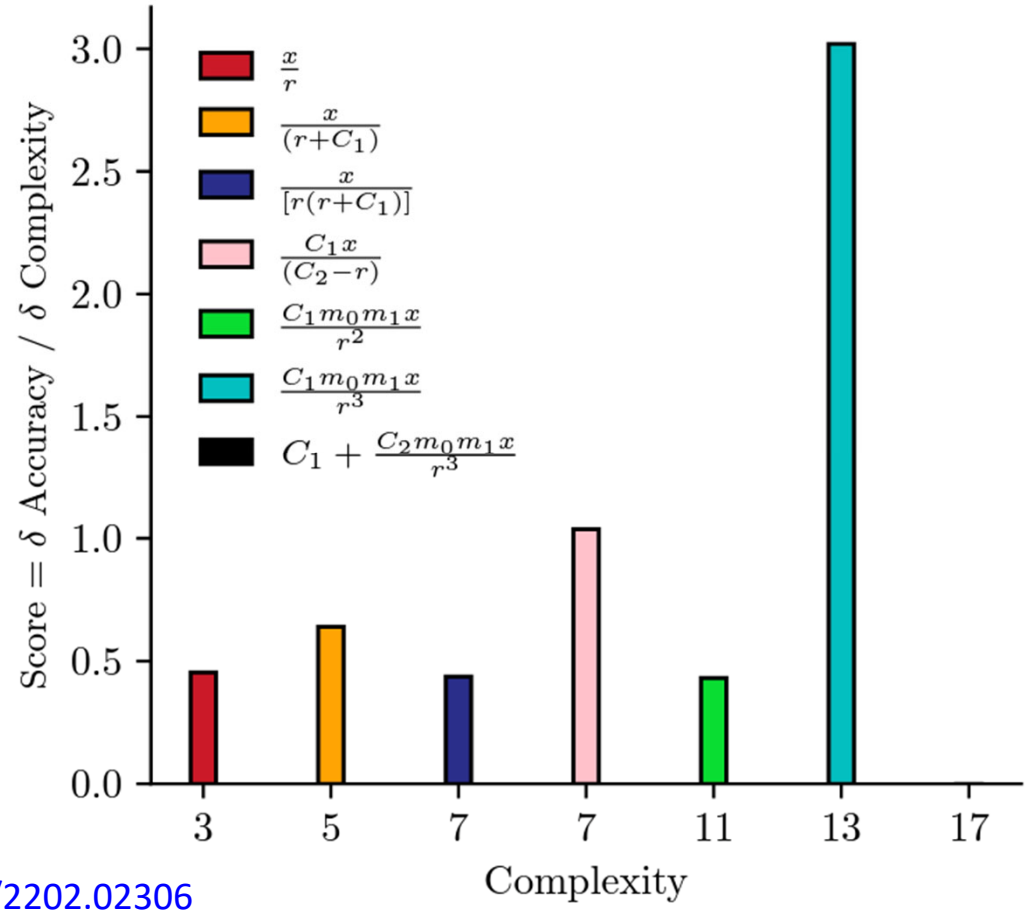


Figure 5: Visualization of all the data in EmpiricalBench, an SR benchmark for science. Color is used to denote additional variables in the cases of relations which depend on more than two inputs. Original data in the discovery of each law is used where easily available. Otherwise, data is generated from the formula with realistic ranges of variables, with a level of noise applied.

Name	Law
Hubble's law	$v = H_0 D$
Kepler's Third Law	$P^2 \propto a^3$
Newton's law of universal gravitation	$\mathbf{F} = G\frac{m_1 m_2}{r^2}\hat{r}$
Planck's law	$B = \frac{2h\nu^3}{c^2}\frac{1}{e^{h\nu/k_B T}-1}$
Leavitt's Law	$M = \alpha \log_{10}(P) + \delta$
Schechter function	$n = \frac{\phi_*}{L_*}(\frac{L}{L_*})^\alpha e^{-\frac{L}{L_*}}$
Bode's law	$a = 0.4 + 0.3(2^n)$
Ideal gas law	$P = \frac{nRT}{V}$
Rydberg formula	$\frac{1}{\lambda} = R_H(\frac{1}{n_1^2} - \frac{1}{n_2^2})$

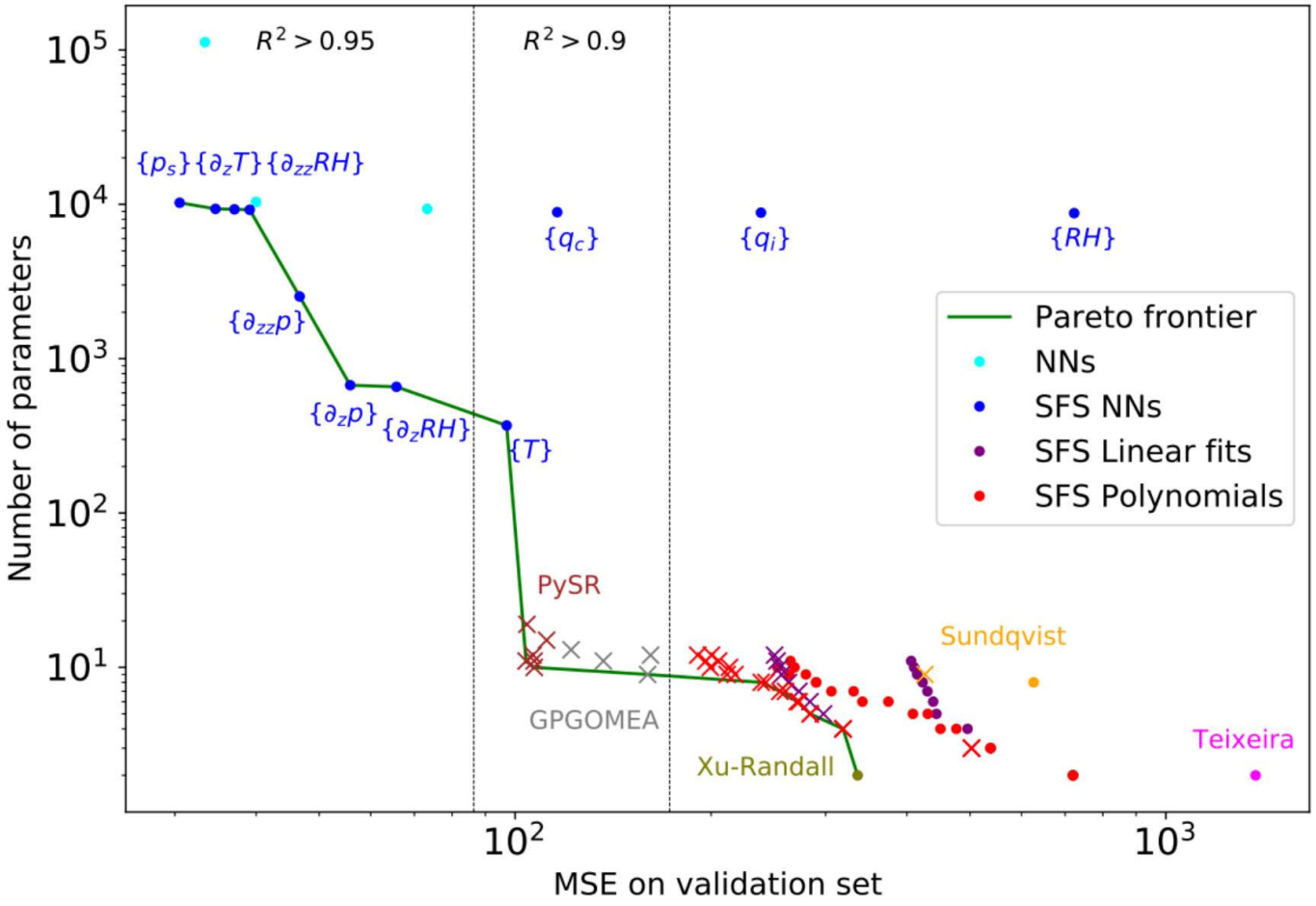
https://arxiv.org/pdf/2305.01582.pdf

Finding the right expression



https://arxiv.org/abs/2202.02306

Finding the right expression



https://arxiv.org/abs/2304.08063