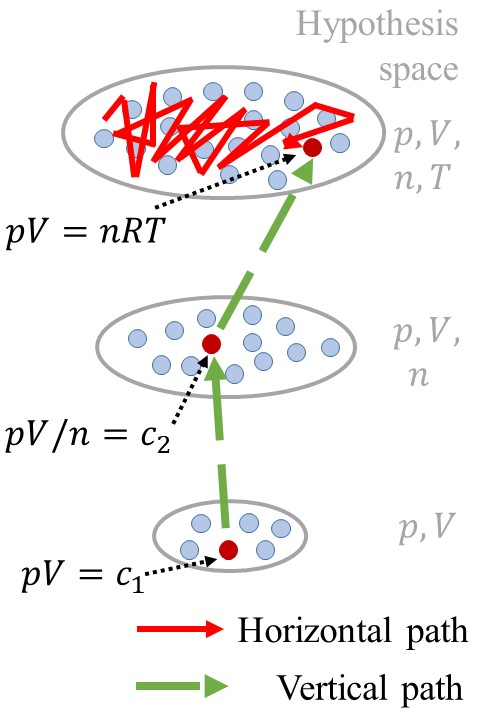
**Symbolic Regression Using Scientific Approaches (SRSci)**

****Automating scientific discovery has been a grand goal dating back to the founders of AI (Herbert Simon et. al. [14, 12, 30]) but remains a holy grail today. The underlying societal impact is immense because of its multiplier effect on applicational domains.

Symbolic regression discovers an underlying equation that maps from the input to the output of a given dataset. It is one of the best benchmarking tasks for scientific discovery, as it mimics the process of discovering physics laws from experimental data. Over the years, much effort has been made toward symbolic regression, including search-based methods [13, 15], genetic programming [27, 29, 25, 5], reinforcement learning [21, 26, 20, 21], deep function approximation [19, 2, 23, 22, 17, 31, 3, 7, 1], and integrated systems [28, 10, 9, 16]. Most endeavor focuses on horizontal discovery paths, i.e., they directly search for the best equation in the full hypothesis space involving all independent variables (red path in Figure 1). This path can be challenging because of the exponentially large full hypothesis space.

Interestingly, the widely applied scientific approaches imply vertical discovery paths, which may be more efficient. To discover the ideal gas law pV = nRT, scientists first held n (gas amount) and T (temperature) as constants and find p (pressure) is inversely proportional to V (volume). They then studied the relationship between pV and n, T. This led to a vertical discovery path (green long dashed path in Figure 1). The first few steps of a vertical path can be significantly cheaper than the horizontal path because the searches are in reduced spaces involving a small number of independent variables. As a result, the vertical discovery has the potential to supercharge state-of-the-art approaches in modeling complex scientific phenomena with more interlocking contributing factors or processes than what current approaches can handle.

This competition calls for **efficient symbolic regressors, which are free to follow vertical (scientific approach motivated), horizontal (machine learning driven), or any new scientific discovery paths.** Essentially, the participants are free to determine what training data is needed to uncover the underlying symbolic expression.

**Design Your Experiments!**

In order to discover a symbolic equation , we will give you an oracle to query. For a minibatch of X that you come up with, the oracle will return , corrupted by a certain level of noise. X is arranged as a matrix, where the i-th row contains the input . should be a vector, where y[i] denotes the i-th output. You are allowed to query the oracle as many times as possible within the time limit.

**Details of the Competition.**

You will need to submit a Python program for symbolic regression. Unlike most symbolic regression programs, the data needs to be obtained from the oracle described above. The oracle is provided to you via an object of the class Equation\_evaluator. Within a given time limit, you are allowed to query the oracle as many times as possible. The way to query the oracle is discussed in [page of how to participate].

Your program needs to write the symbolic equation you find into a pre-specified output file. The program is allowed to overwrite the output file as many times as you’d like within the time limit. Only the equation in the last non-overwritten output file will be evaluated.

**Competition Tracks.**

[Provide a table of time limit, noise type, dataset]

**How to Participate?**

[Go to the page of how to participate]

**How We Will Evaluate.**

We will randomly generate a minibatch of input X, and obtain the ground-truth output , and your prediction . Here is the symbolic equation you wrote in the output file. We will report the symbolic equation complexity (measured as the number of nodes in the expression tree), and various loss functions between and . All the reported values will be the median value across all the equations considered in a single competition track.

Here is the [page] describing all the loss functions.

Here is the [page] describing our computational environment of evaluation.

**Organizer**

* Nan Jiang (Purdue University), jiang631 at purdue dot edu
* Yexiang Xue (Purdue University), yexiang at purdue dot edu

**Discord Contacts**

If you want to ask any question, provide some feedback or simply chit-chat, join us at the [Discord server](https://discord.gg/MeGnkHr4).