Symbolic Regression using Mixed-Integer Nonlinear Optimization

Vernon Austel [1] Cristina Cornelio [1] Sanjeeb Dash [1] João Gonçalves [1] Lior Horesh [1] Tyler Josephson [2]
Nimrod Megiddo [3]

Abstract

The Symbolic Regression (SR) problem, where the goal is to find a regression function that does not have a pre-specified form, but is any function that can be composed of a list of operators, is a hard problem in machine learning, both theoretically and computationally. Genetic programming based methods, that heuristically search over a very large space of functions, are the most commonly used methods to tackle SR problems. An alternative mathematical programming approach, proposed in the last decade, is to express the optimal symbolic expression as the solution of a system of nonlinear equations over continuous and discrete variables that minimizes a certain objective, and to solve this system via a global solver for mixed-integer nonlinear programming problems. Algorithms based on the latter approach are often very slow. We propose a hybrid algorithm that combines mixed-integer nonlinear optimization with explicit enumeration and incorporates constraints from dimensional analysis. We show that our algorithm is competitive, for some synthetic data sets, with a state-of-the-art SR software and a recent physics-inspired method called AI Feynman.

1. Introduction

Regression methods have numerous applications in science and economics. In traditional regression methods, such as linear or logistic regression, the relationship between a dependent variable – say y – and a number of independent variables – say $x_1, \dots, x_n$ – is assumed to have a fixed, parametric, functional form, and the parameters are calculated from data to minimize a specific "loss" function. For example, in linear regression, the functional form is assumed to be a linear function of the independent variables: $y = a_1x_1 + \cdots + a_nx_n$. The unknown parameters a_i are the coefficients that are calculated so as to minimize, say, the least-square error: $\sum_i (y^i - \sum_j a_j x_j^i)^2$, where y^i and x_j^i are the values of y and x_j, respectively, in the ith data point.

In symbolic regression (SR), both the functional form of of a regression function, and the associated parameters or coefficients are calculated from data so as to minimize some loss function. The functional form is assumed to be anything that can be composed from a given list of basic functions or operators applied to the independent variables and arbitrary constants. For example, if the operators are $+$, $-$ and $\times$, then the space of all possible functions is the set of all polynomials of arbitrary degree.The usefulness of SR was demonstrated in (Connor & Taylor, 1977; Langley, 1981; Willis et al., 1997; Davidson et al., 2003; Schmidt & Lipson, 2009; 2010).

Given its generality, symbolic regression is clearly a difficult problem and there has been a lot of research into designing computationally effective algorithms. Starting with (Koza, 1992), SR problems were solved with genetic programming (GP) (Augusto & Barbosa, 2000; Banzhaf et al., 1998), and SR is often treated as being synonymous with GP (Stephens, 2019). In (Korns, 2011), the accuracy of SR solutions achieved by genetic algorithms was claimed to be poor. "Bloat" is another common issue in GP based methods (obtaining functions with high description length).

Some recent research on improving symbolic regression techniques is aimed at speeding them up (Luo et al., 2017), finding accurate constants in the derived symbolic mathematical expression (Izzo et al., 2017), and searching for implicit functional relationships (Schmidt & Lipson, 2010). These approaches are primarily based on genetic algorithms. Another approach explicitly populates a large hypothesis space of functions, on which sparse selection is applied (Brunton et al., 2016). In (Udrescu & Tegmark, 2019), a non-GP SR algorithm is developed , potentially well-suited for physics problems, that combines neural-network fitting with several techniques inspired by physics, for example, dimensional analysis, and also complete enumeration of symbolic expressions of a certain limited size.

[1] IBM Research AI, 1101 Kitchawan Road, Yorktown Heights, NY 10598 [2] Department of Chemistry, University of Minnesota [3] IBM Research - Almaden, 650 Harry Rd, San Jose, CA 95120. Correspondence to: Sanjeeb Dash <sanjeebd@us.ibm.com>.

A major challenge in symbolic regression is the difficulty of finding scientifically meaningful models out of the large number of possible models that may fit given data. In (Schmidt & Lipson, 2009; 2010), partial derivatives are matched, in addition to fitting data, to discover meaningful functions. In AI Feynman (Udrescu & Tegmark, 2019), dimensional analysis is used to both speed up the solution process, and to generate meaningful physical models.

Some recent papers proposed mathematical optimization based approaches for obtaining regression models: a model is obtained as a solution of a constrained optimization problem that selects the kind of model and optimizes its parameters. Various Mixed-Integer Programming (MIP) formulations were proposed in (Bertsimas & Shioda, 2007) for finding classification and regression models. Mixed-Integer Nonlinear Programming (MINLP) formulations for SR were introduced in (Cozad, 2014), (Avron et al., 2015) and (Horesh et al., 2016). Some computational results were presented in (Cozad, 2014; Austel et al., 2017; Cozad & Sahinidis, 2018). In these three papers, the BARON (Tawarmalani & Sahinidis, 2005) solver was used to obtain globally optimal (see the next section) solutions, though other similar solvers can be used. An advantage of the MINLP framework is that, in principle, globally optimal symbolic expressions can be obtained along with certificates of optimality. However, existing MINLP models for SR, and the solvers for such problems, have limited scalability. In addition, the MINLP framework allows one to incorporate domain knowledge that can be encoded via nonlinear constraints: adding constraints to enforce dimensional consistency is proposed in (Gunawan et al., 2017).

1.1. Our contribution

In this paper, we develop a symbolic regression algorithm based on solving MINLPs inspired by and extending the free-form SR discovery framework (Austel et al., 2017) that was tested on a few toy problems, such as relations found by Kepler and Galileo. Our work is different from the work in the previous paragraph in multiple ways. Firstly, we sacrifice global optimality to get a faster algorithm (thus our algorithm can be viewed as a heuristic, though we can, in principle obtain globally optimal solutions given enough time). Secondly, we choose a different MINLP formulation of an expression tree than in previous papers, and also employ a different search strategy. Finally, we add constraints to enforce dimensional consistency.

We compare our method with AI Feynman (Udrescu & Tegmark, 2019) on the synthetic, physics-inspired data set in the associated paper, and show that our MINLP-based approach is still competitive for a number of problems, and obtains solutions to some problems that are not solved by Eureqa (but solved by AI Feynman). We can also obtain solutions to some problems from less data.

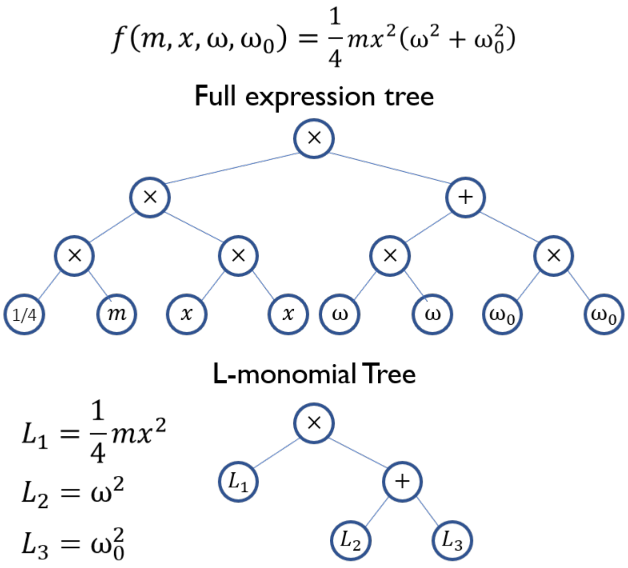

Figure 1. Expression tree for $\frac{1}{4}mx^2(\omega^2 + \omega_0^2)$, expressed with full arithmetic notation, and as a more compact tree of L-monomials.

2. Related work

A symbolic regression scheme consists of a space of valid mathematical expressions composable from a basic list of operators (we assume these to be unary or binary), and a mechanism for exploring the space. Each valid mathematical expression can be represented by an expression tree: a rooted binary tree where each non-leaf node has an associated binary or unary operator ($+$, $-$, $\times$, $\sqrt{}$, log, etc.), and each leaf nodes has an associated constant or independent variable. An example of an expression tree for the expression

$$\frac{1}{4}mx^2(\omega^2 + \omega_0^2)$$

is presented in Figure 1. The second tree in the figure is explained later.

A typical genetic programming (GP) solver searches the space of expressions using a genetic algorithm The Eureqa package (Schmidt & Lipson, 2014), based on the article (Schmidt & Lipson, 2009), is a state-of-the-art GP-based solver. Another popular solver is `gplearn` (Stephens, 2019). AI Feynman (Udrescu & Tegmark, 2019) first trains a neural network (NN) on the input data, so that the function encoded by the NN is an accurate fit for the input data. It then uses tests if the true function has various properties such as separability, symmetry etc, by evaluating the NN at points not originally in the data set. In addition, it uses dimensional analysis to speed up the search for the best symbolic expression. In (Udrescu & Tegmark, 2019), the authors applied their algorithm to one-hundred physics equations, and claim to rediscover all of them. They observe that Eureqa obtained solutions to only 71 of these instances.

2.1. Globally optimal symbolic regression

In a series of works (Cozad, 2014; Cozad & Sahinidis, 2018; Avron et al., 2015; Horesh et al., 2016; Austel et al., 2017), symbolic regression (SR) methods based on combined discrete and continuous optimization were developed, and the symbolic regression problem was formulated in various ways as a Mixed-Integer Nonlinear Programming (MINLP) problem. The MINLP problem is solved to global optimality using an off-the-shelf MINLP solver such as BARON (Tawarmalani & Sahinidis, 2005), COUENNE or SCIP (Maher et al., 2017). These solvers solve problems of the form

$$
\begin{aligned}
\min \quad & f(x,y) && (1)\\
s.t. \quad & g(x,y) \le b && (2)\\
& x \in \mathbb{R}^m, y \in \mathbb{Z}^m && (3)
\end{aligned}
$$

where x is a vector of m continuous variables, y is a vector of n discrete variables, $f(x,y)$ is a real-valued function that can be composed using a finite list of operators such as $+, -, \times, /, exp(\cdot)$ etc. (this list will vary with each solver), and $g(x,y)$ is a vector-valued function created using the same operators. These solvers use various convex-relaxation schemes to obtain lower bounds on the objective-function value, and utilize these bounds in branch-and-bound schemes to obtain globally optimal solutions.

This approach produces a globally-optimal mathematical expression (along with a certificate of optimality) while avoiding an exhaustive search of the solution space. Another advantage is that it directly produces correct, within a tolerance, real-valued constants; most other methods use specialized algorithms to refine constants (Izzo et al., 2017) and cannot guarantee global optimality.

In these MINLPs, the set of valid binary expression trees is specified by a set of constraints over discrete and continuous variables. The discrete variables of the formulation are used to define the structure of the expression tree including the assignment of operators to non-leaf nodes and whether a leaf node is assigned a constant or a specific independent variable. The continuous variables are used for the undetermined constants, and also to evaluate the resulting symbolic expression for specific numerical values associated with individual data points. The objective functions vary from accuracy (measured as sum of squared deviations) to model complexity in (Austel et al., 2017).

In (Austel et al., 2017), for example, the MINLP has the following form:

$$
\begin{array}{lcl}
\min & \mathcal{C}(f_\theta) & \text{complexity}\\
\text{s.t.} & \theta \in \mathcal{T} & \text{grammar}\\
& v_i = f_\theta(x^{(i)}), \ \forall i \in I & \text{prediction}\\
& \mathcal{D}(f_\theta(x), y) \le \varepsilon & \text{error}
\end{array}
$$

where f_θ represents an expression tree defined by the structural and continuous decision variables collectively designated as θ; $\mathcal{T}$ is the universe of valid expression trees; $\mathcal{C}$ measures the description complexity; $\mathcal{D}$ measures error of the predicted values v; y is the vector of observations for inputs $\{x^{(i)}\}_{i \in I}$.

3. System Description

We first describe the basics of our system (without dimensional analysis), and then later explain how we incorporate dimensional analysis.

We take as input a list of operators (for this discussion, assume the input operators are $+, -, \times, /$, and $\sqrt{}$), an upper bound d on the tree depth, an upper bound k on the number of constants, and a domain for the constants $[-\Omega, \Omega]$. In Figure 1, there is a single constant (distinct from 1) with value $1/4$ in the expression tree and in the L-monomial tree. In the prior work on globally optimal symbolic regression, the MINLP formulations typically have discrete variables that (1) determine the placement of the operators in nodes of the expression tree and (2) the mapping of independent variables to leaf nodes, and (3) determine whether to map an independent variable or a constant to a leaf node.

In our formulation, we simply do not have discrete variables for (1). We explicitly enumerate all possible assignments of the operators in expression trees up to depth d. More precisely, if a“partial” expression tree is one where the leaf nodes are undetermined, but the operator assignments to non-leaf nodes are determined, then we enumerate all possible partial expression trees up to a certain depth. Our assignment of variables/constants to leaf nodes is also different from prior work. Let $x_1, \ldots, x_n$ be all the independent variables. Instead of assuming that each leaf node is either a constant h or one of $x_1, \ldots, x_n$ as in the prior work, we assume each leaf node is a one-term multivariable Laurent polynomial of the form

$$h x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}, \tag{4}$$

where $a_1, \ldots, a_n$ are (undetermined) integers (for computational efficiency, we limit these integers to lie in the range $[-\delta, \delta]$ for some input constant δ), and h represents an (undetermined) constant in the final expression. We refer to an expression of the type (4) an *L-monomial*. In other words, rather than assigning a single variable or constant to a leaf node, we potentially assign both variables and constants, and also multiple variables (with positive or negative powers to a leaf node). We call the resulting trees *generalized expression trees*, and *gentrees* for convenience.

Each gentree T (with depth say d) corresponds to a symbolic expression: each non-leaf node with height 1 corresponds to the expression formed by applying the operator at the node to the expressions (i.e., L-monomials) in the children nodes, and non-leaf nodes at greater heights are handled in the same manner in order of height. The only gentree with depth 0 corresponds to the L-monomial L_1, whereas the gentrees of depth 1 correspond to the expressions $\sqrt{L}$, $L_1 + L_2$, $L_1 \times L_2$, L_1/L_2 and $L_1 - L_2$, respectively, where L_1 and L_2 are L-monomials.

3.1. Pruning the list of gentrees

We try to reduce the number of relevant partial expression trees/gentrees by removing a number of "redundant" trees using the fact that the set of nonzero (the constant h in 4 is nonzero) L-monomials is closed under multiplication and division. Notice that both $L_1 \times L_2$ and L_1/L_2 are L-monomials and thus can be represented by a single expression L_1. In other words, if there is a symbolic expression f of the form $f = L_1 \times L_2$ that fits our data, then there is one of the form $f = L_1$. Accordingly our first few pruning rules are: remove a tree T from the list $\mathcal{T}$ of all gentrees with depth up to d if T has the subexpression

[R1] $L_1 \times L_2$ or L_1/L_2.

[R2a] $L_1 \times (L_3 \pm L_4)$.

[R2b] $(L_1 \pm L_2) * (L_3 \pm L_4)$.

[R3] $(L_1 \pm L_2)/L_3$.

The first rule was explained above. The second follows by associativity: $L_1 * (L_3 + L4) = L_1 \times L_3 + L1 \times L_4 = L_1' + L_2'$, for some L-monomials L_1' and L_2'. If a T with a subexpression $L_1 * (L_3 + L4)$ appears in $\mathcal{T}$, then replacing this subexpression by $L_1' + L_2'$ results in another gentree T' of the same depth, which must therefore be in $\mathcal{T}'$. The same idea can be applied to the subexpression $L_1 * (L_3 - L_4)$. We can apply associativity twice to justify rule R2b, as $(L_1 + L_2) * (L_3 + L_4) = L_1' + L_2' + L_3' + L_4'$, for some L_i' (the same argument holds when either $+$ is replied by a $-$ in R2b). Once again, the second expression has the same depth as the first, and therefore there must be a tree in $\mathcal{T}$ containing it. R3 can be explained similarly.

3.2. MINLP formulation for a gentree

Let the data points be $x^{(i)}$ for $i \in I$, where I is an index set, and let the features/independent variables be $x_1, \dots, x_n$. Let y stand for the observations/dependent variable, with $y^{(i)}$ standing for the ith observation (of the dependent variable). Let T be a given gentree with m leaf nodes, and let p be the vector of all integer variables corresponding to the powers of the independent variables in the different leaf nodes. Then $p \in \mathbb{Z}^{mn}$. Let $h_1, \dots h_m$ be the variables corresponding to the constants in leaf nodes $1, \dots, m$. Finally, assume we have a 0-1 variable z_i that determines whether or not the ith leaf node has a constant h_i that is different from one. Thus the (vector) variables in our model are p, z and h. Let $f_{h,p,z,T}$ stand for the symbolic expression defined by fixing the values of these variables. Then the MINLP we solve can be framed as

$$\min \quad \textstyle\sum_{i \in I} (y^{(i)} - f_{h,p,z,T}(x^{(i)}))^2 \tag{5}$$

$$\text{s.t.} \quad -\delta \le p_i \le \delta \quad \text{for } i = 1, \dots, mn \tag{6}$$

$$-\Omega z_i + (1 - z_i) \le h_i \le \Omega z_i + (1 - z_i) \quad \text{for } i = 1, \dots, m \tag{7}$$

$$\textstyle\sum_{i=1}^{m} z_i \le k \tag{8}$$

$$z \in \{0,1\}^m, \;\; p \in \mathbb{Z}^{mn} \tag{9}$$

The constraints (9) and (6) force z_i to take on 0-1 values, and p_i to take on values in the range $\{-\delta, -\delta + 1, \dots, \delta\}$. The constraint (7) restricts h_i to lie in the range $[-\Omega, \Omega]$ if z_i has value 1, and forces h_i to take on value 1, when z_i has value 0. The constraints (8) allow at most k of the z_i variables to have value 1, and therefore at most k of the h_i values to be different from 1. The function $f_{h,p,z,T}$ is composed of the operators in the non-leaf nodes of T from the L-monomials in the leaf nodes. The objective function is the sum of squares of differences between the symbolic expression values for each input data point and the corresponding dependent variable value (call it the *least-square error*).

We use BARON to solve the MINLPs we generate, and thus T and $f_{h,p,z,T}$ are limited by the operators that BARON can handle $(+, -, \times, /, \exp(), \log())$. We will illustrate f for a few examples, rather than specify it formally. Suppose we are trying to derive the formula $F = G\,\frac{m_1 m_2}{r^2}$, where m_1, m_2 and r are independent variables, and G is an unknown constant, and we have multiple data points (for $i \in I$) with values for the independent variables, and for associated F. If T is a depth 0 gentree, then T is just a single L-monomial, and $f_{h,p,z,T} = h m_1^a m_2^b r_i^c$ where a, b, c are undetermined integers in the range $[-\delta, \delta]$, and h is an undetermined real number in the range $[-\Omega, \Omega]$. The objective function is then

$$\sum_{i \in I} (F_i - h m_{1i}^a m_{2i}^b r_i^c)^2 \; ,$$

where F_i, m_{1i}, m_{2i} and r_i are the values of F, m_1, m_2, r in the ith data point. If the gentree can be written as $L_1 + L_2$, then the objective function becomes

$$\sum_{i \in I} (F_i - (h_1 m_{1i}^{a_1} m_{2i}^{b_1} r_i^{c_1} + h_2 m_{1i}^{a_2} m_{2i}^{b_2} r_i^{c_2}))^2;$$

here a_i, b_i, c_i and h_i are undetermined, and we solve for these values.

As depicted in Figure 1, an expression tree of a certain depth can sometimes be represented by a gentree of smaller depth. Therefore, by enumerating gentrees of a certain depth, one obtains a much richer class of functions than can be obtained by expression trees of the same depth.

3.3. Enumeration and parallel processing

By enumerating the generalized expression trees and solving them separately, the problem is divided into multiple, easier-to-solve sub-problems (operator placement in non-leaf nodes does not have to be determined any more) that can be solved much more quickly. This "divide and conquer" formulation can obtain solutions to problems that were intractable for the formulation in (Austel et al., 2017).

When available, we exploit parallelism, and run multiple threads, with one MINLP corresponding to a single gentree in a single thread. If we obtain a solution that fits the data within a prescribed tolerance from a gentree with depth d', then we stop all processes/threads containing gentrees with depth $> d'$. We terminate a gentree if the lower bound on the least-square error exceeds the least-square error found from other gentrees of the same or lower depth. Thus we execute a branch-and-bound type search, and implicitly search for the least depth gentree that fits the data.

If we have more gentrees than available cores, we process them in a round robin fashion; if t is the number of cores, we start solving for t gentrees in parallel, and after a fixed amount of time (10 seconds), we pause the first t gentrees, and start solving t more, till we either find a solution or run out of gentrees in which we case we start from the first gentree. The gentrees are sorted by a measure of complexity (roughly equal to the number of nodes). Of course, the gentree enumeration (and thus our overall algorithm) grows exponentially with gentree depth d': the number of gentrees for $d' = 2, 3, 4, 5$ are, respectively, 7, 60, 4485 and over 100,000. Our gentree enumeration approach is unlikely to be tractable for $d' >= 6$ and is already hard for $d' = 4$ if we include more operators than listed earlier. But each L-monomial can represent large expression trees.

3.4. Dimensional analysis

Consider Newton's law of universal gravitation which says that the gravitational force between two bodies is proportional to the product of their masses and inversely proportional to the square of the distance between their centers (of gravity):

$$F \propto \frac{m_1 m_2}{r^2}$$

and the constant of proportionality is G, the gravitational constant. Therefore, we have

$$F = G\,\frac{m_1 m_2}{r^2}.$$

The units of G are chosen so that units of the right-hand-side expression equal the units of force (mass × distance / time-squared). Suppose one is given a data set for this example, where each data item has the masses of two bodies and the distance and gravitational force between them. Dimensional analysis would rule out $F = m_1 m_2 / r^2$ or $F = m_1/m_2 + m_2 r$, for example, as possible solutions.

If constants can have units and every L-monomial can have a such a constant, then dimensional analysis conveys essentially no information. For example, we can choose constants h_1, h_2, h_3 with appropriate dimensions so that both the expressions $F = h_1 m_1 m_2 / r^2$ and $F = h_2 m_1/m_2 + h_3 m_2 r$ satisfy all dimensional requirements. In the AI Feynman experiments, the authors do not allow constants to have units, and also ensure that for all required universal constants that have units (such as the Gravitational constant), both numerical values and units are given as input. We allow constants to have units or not based on an input flag.

We next explain the constraints we add to our formulation to enforce dimensional consistency. Assume that all constants have no units, and that $k = 2$, and $\delta = 2$, and $d = 1$. For the gravitation example (without the gravitational constant G as an input), each L-monomial L has the form

$$L = h m_1^a m_1^b r^c, \tag{10}$$

where a, b and c are bounded integer variables with an input range of $[-2, 2]$, and h is a variable representing a constant. We add to the system of constraints (6) - (9) linear constraints that equate the units of the symbolic expression to those of the dependent variable. For the case our gentree consists of a single node (as in the first tree in Figure 1), our symbolic expression has the form (10), and we add the linear constraints

$$a \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + b \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + c \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}.$$

Here, each component of a column vector corresponds to a unit (mass, distance, time, respectively). The column on the right-hand side represents the units of force, i.e., mass times distance divided by squared time. The columns on the left-hand side represent the dimensionalities of mass and distance, respectively. The third of the above linear equations does not have a solution.

Next, assume the symbolic expression is $L_1 + L_2$ where $L_1 = h_1 m_1^{a_1} m_2^{b_1} r^{c_1}$ and $L_2 = m_1^{a_2} m_2^{b_2} r^{c_2}$, and the unkowns are a_i, b_i, c_i and h_i. The units of L_1 and L_2 have to match, and must also equal the units of F. The linear constraints we add are

$$a_1 \begin{bmatrix}1\\0\\0\end{bmatrix} + b_1 \begin{bmatrix}1\\0\\0\end{bmatrix} + c_1 \begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}1\\1\\-2\end{bmatrix} \tag{11}$$

$$a_2 \begin{bmatrix}1\\0\\0\end{bmatrix} + b_2 \begin{bmatrix}1\\0\\0\end{bmatrix} + c_2 \begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}1\\1\\-2\end{bmatrix}. \tag{12}$$

Finally, for the symbolic expression $L_1 \times L_2$, to compute the units of this expression we need to add up the units of L_1 and of L_2. Thus we sum the left hand sides of (11) and (12) and equate this sum to the righ hand side of (11). We can similarly deal with dimension matching in the remaining gentrees of depth 1 via linear constraints. For greater depth gentrees, we apply the ideas above to depth 1 (non-leaf nodes), and then to depth 2 nodes and so on.

4. Results

4.1. Feynman Database for Symbolic Regression

In our first experiment, we choose a subset of the 100 problems from the Feynman Database for Symbolic Regression (FSReD), created in (Udrescu & Tegmark, 2019). We first eliminate the 19 problems which involve operators our system cannot handle because these operators cannot be handled by the underlying MINLP solver we use (namely $\sin, \cos, \arcsin, \tanh$). We give detailed results on 32 out of the remaining 81 instances with varied properties in Table 1: some can easily be solved by AI Feynman and Eureqa, and some are hard for these solvers, and some need many data points or limited noise.

We set the depth limit d to 3, and the number of constants k to 1. We terminate when we find an expression with objective value (squared error) less than 10^{-4} (error tolerance). The operators we use are described in Section 3. We set Ω to 100, so all constants are in the range $[-100, 100]$. We set $\delta = 2$ for computational efficiency. This means that each power in an L-monomial lies in the range $[-2, 2]$. Under this setting, the pruning rules – R1, R2,R3 – remove potentially non-redundant gentrees, and our algorithm does not explore all possible symbolic expressions that are representatable as a gentree of depth up to 3. Though L-monomials are closed under multiplication, L-monomials with bounded powers are not. For example, if there is a solution of the form $x_1^3 x_2^2$, we may not find it.

We constrain the search in three more ways. We bound the sum of the variable powers in an L-monomial by an input number $\tau = 6$. Thus, for each term of the type (4), we add the constraint $\sum_{i=1}^n |a_i| \leq \tau$, which is representable by a linear constraint using n auxiliary variables $a'_i : \sum_{i=1}^n a'_i \leq \tau$ and $a'_i \geq 0, -a_i \leq a'_i$ and $a_i \leq a'_i$ for $i = 1, \ldots, n$. Secondly, we add another pruning rule. We remove any gentree which contains a subexpression of the form $\sqrt{L}$ where L is an L-monomial (we allow $\sqrt{L_1 + L_2}$, for example). Finally, we drop the operator $-$ from our set. However, as we allow positive and negative constants, we can simulate a $-$ operator. But because we only allow at most one constant, this means that we effectively allow a single $-$ operator in our generated solutions. All constants are allowed to have units.

We give our results in Table 1. The first column contains the equation label from FSReD, the second shows the actual function. The third column gives the running time (in seconds) of our code on a machine with 30 cores, and with $\tau = 6$. A 'f' in this column implies our code did not produce the correct answer within the time limit of 600 seconds. The fourth column gives the outcome of our experiment with noise. The "data used" column contains the number of data points we use, which is (the first) 10. Our method does not scale well with the number of data points: equation (5) becomes more complex, and the resulting MINLPs harder. We can solve for 20-30 points for most instances, but the running time increasing significantly for some problems. In columns 6-8 we give the time taken by AI Feynman, the min number of points it needs to get the correct answer, and the maximum noise level under which it obtains the correct answer, all taken from the associated paper, as is the Eureqa solution status in the last column.

We fail on 5 instances, but on the remaining instances, our results are mixed in comparison to AI Feynman. We are often slower, but sometimes faster (II.24.17, II.38.3, III.10.19), even if we scale our times by 30 (to account for the parallelism used by our code). We note that AI Feynman searches over more operators, and thus a potentially larger search space. For three instances – I.48.20, II.11.27, III.10.19 (with the number of data points in bold) – we obtain the correct answer with one-tenth the number of data points needed by AI Feynman.

To solve I.6.20a, indicated by a '*', we first spend the time limit of 600 seconds using the default set of operators, and then restart with $\sqrt{\ }$ replaced by $\exp()$, and then take another 100 seconds. This is why we report a running time of 700. While using $\exp()$, we also decrease the error tolerance to 10^{-8}. (We note that AI Feynman also cycles through different sets of operators). Out of the remaining 49 problems, we are able to solve 36 instances, and fail on 13. Changing some of our parameters will allow additional problems to be solved at the cost of making the code slower. For example, increasing the number of constants k to 2 allows our code to solve I.15.3t. Finally, BARON can handle the $\log()$ operator, but we have not experimented with it.

In a second experiment, we set the relative noise level to 10^{-2} (as described in the AI Feynman paper), and report

Eqn. label	Eqn.	exp 1	noise exp	data used	AIF time	AIF min data	AIF max noise	Eureqa solved
I.6.20a*	$e^{-\theta^2/2}/\sqrt{2\pi}$	700		10	16	10	10^{-2}	no
I.9.18	$\frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	f		10	5975	10^6	10^{-5}	no
I.10.7	$\frac{m_0}{\sqrt{1-v^2/c^2}}$	11		10	14	10	10^{-4}	no
I.12.2	$\frac{q_1q_2}{4\pi\varepsilon r^2}$	1	yes	10	17	10	10^{-2}	yes
I.12.4	$\frac{q_1}{4\pi\varepsilon r^2}$	1		10	12	10	10^{-2}	yes
I.13.4	$\frac{1}{2}m(v^2+u^2+w^2)$	6		10	22	10	10^{-4}	yes
I.13.12	$Gm_1m_2(\frac{1}{r_2}-\frac{1}{r_1})$	13		10	20	10	10^{-4}	yes
I.15.3t	$\frac{t-ux/c^2}{\sqrt{1-u^2/c^2}}$	f		10	20	10^2	10^{-4}	no
I.15.10	$\frac{m_0v}{\sqrt{1-v^2/c^2}}$	2		10	13	10	10^{-4}	no
I.16.6	$\frac{u+v}{1+uv/c^2}$	10	**yes**	10	18	10	10^{-3}	no
I.18.4	$\frac{m_1r_1+m_2r_2}{m_1+m_2}$	8		10	17	10	10^{-2}	yes
I.24.6	$\frac{1}{4}m(\omega^2+\omega_0^2)x^2$	3		10	22	10	10^{-4}	yes
I.25.13	q/C	1		10	10	10	10^{-2}	yes
I.27.6	$\frac{1}{\frac{1}{d_1}+\frac{n}{d_2}}$	5	yes	10	14	10	10^{-2}	yes
I.32.5	$q^2a^2/(6\pi\varepsilon c^3)$	f		10	13	10	10^{-2}	yes
I.34.10	$\frac{\omega_0}{1-v/c}$	4		10	13	10	10^{-3}	no
I.34.14	$\frac{1+v/c}{1-v^2/c^2}\omega_0$	7		10	14	10	10^{-3}	no
I.39.11	$\frac{1}{\gamma-1}p_FV$	3		10	13	10	10^{-3}	yes
I.43.43	$\frac{k_bv}{(\gamma-1)A}$	5		10	16	10	10^{-3}	yes
I.48.20	$\frac{mc^2}{\sqrt{1-v^2/c^2}}$	8		**10**	108	10^2	10^{-5}	no
II.2.42	$k(T_2-T_1)A/d$	13	**yes**	10	54	10	10^{-3}	yes
II.11.3	$\frac{qE_f}{m(\omega_0^2-\omega^2)}$	8	**yes**	10	25	10	10^{-3}	yes
II.11.20	$\frac{n_\rho p_d^2E_f}{3k_bT}$	9		10	18	10	10^{-3}	yes
II.11.27	$\frac{n\alpha}{1-n\alpha/3}\epsilon E_f$	36	**yes**	**10**	337	10^2	10^{-3}	no
II.11.28	$1+\frac{n\alpha}{1-n\alpha/3}$	f	**yes**	10	1708	10^2	10^{-4}	no
II.21.32	$\frac{q}{4\pi\varepsilon r(1-v/c)}$	13		10	21	10	10^{-3}	yes
II.24.17	$\sqrt{\frac{\omega^2}{c^2}-\frac{\pi^2}{d^2}}$	**2**	**yes**	10	62	10	10^{-5}	no
II.34.11	$g\_qB/(2m)$	1		10	16	10	10^{-4}	yes
II.36.38	$\frac{\mu_mB}{k_bT}+\frac{\mu_m\alpha M}{\varepsilon c^2k_bT}$	f		10	77	10	10^{-2}	yes
II.37.1	$\mu_M(1+\chi)B$	1		10	15	10	10^{-3}	yes
II.38.3	$\frac{YAx}{d}$	**1**	**yes**	10	47	10	10^{-3}	yes
III.10.19	$\mu_M\sqrt{B_x^2+B_y^2+B_z^2}$	**6**	**yes**	**10**	410	10^2	10^{-3}	yes

Table 1. Results on a subset of problems from the Feynman Database for Symbolic Regression

a "yes" if one of the gentrees returns the expected formula as a solution and has low error. When the "yes" is in bold, AI Feynman needs a lower noise level. However, as we generate arbitrary constants, for some problems we find solutions that have lower squared error than the true formula (on the 10 data points we chose), and one should apply AIC or BIC to choose from alternate formulae.

A major difference between our code and AI Feynman is that our code finds arbitrary constants, whereas AI Feynman finds combinations of fixed symbols (such as small integers and π and other known physical constants). For example, in I.12.2, we find $1/(4\pi)$ as 0.07957747. For problem I.13.12, AI Feynman uses the known value of Newton's Gravitational constant and cannot discover it, whereas our code can.

5. Conclusions

A few MINLP based SR methods have been proposed earlier, but these are not very scalable. These methods obtain an "optimal" solution and a certificate of optimality. We proposed an MINLP based approach that enumerates expression trees with fixed operators, and uses MINLP to solve for the form of the leaves. Our approach is competitive with state-of-the-art SR packages on some difficult problems. It incorporates dimensional analysis (a first for an MINLP based implementation) but can also find real-valued constants (as can gplearn). A number of drawbacks – such as limitations on number of data points, and high variability with respect to parameters – of our method suggest future directions of research. Even though our overall method is a heuristic, the MINLP solves for individual gentrees return certificates of optimality. One can speedup the method by developing heuristics to find good solutions for a gentree and terminating before proving optimality.

Our code combines a number of useful features: a) it infers results from limited data, b) discovers complex constants, and c) deals with some noise as in real data, and d) is reasonably fast. AI Feynman cannot handle arbitrary constants or much noise, and Eureqa cannot handle explicit equations (such as dimensional equations, though dimensional equations can be handled implicitly by an application of the the Buckingham-Pi and the implied data preprocessing).

Acknowledgements Tyler Josephson was primarily supported by the U.S. Department of Energy (DOE), Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences and Biosciences under Award DE-FG02-17ER16362. Tyler Josephson and Lior Horesh gratefully acknowledge the support of the Institute for Mathematics and its Applications (IMA), where a part of this work was initiated.

References

Augusto, D. A. and Barbosa, H. J. Symbolic regression via genetic programming. In *Proceedings of the Sixth Brazilian Symposium on Neural Networks*, pp. 173–178. IEEE, 2000.

Austel, V., Dash, S., Gunluk, O., Horesh, L., Liberti, L., Nannicini, G., and Schieber, B. Globally optimal symbolic regression. *NIPS Symposium on Interpretable Machine Learning*, 2017.

Avron, H., Horesh, L., Liberti, L., and Nahamoo, D. Globally convergent system and method for automated model discovery, June 30 2015. U.S. Patent Application 20170004231, application No. 14/755,942 filed June 30, 2015.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. *Genetic programming: an introduction*, volume 1. Morgan Kaufmann, San Francisco, 1998.

Bertsimas, D. and Shioda, R. Classification and regression via integer optimization. *Operations Research*, 55:252–271, 2007.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

Connor, J. W. and Taylor, J. B. Scaling laws for plasma confinement. *Nuclear Fusion*, 17(5):1047, 1977.

Cozad, A. *Data- and theory-driven techniques for surrogate-based optimization*. PhD thesis, Carnegie Mellon, Pittsburgh, PA, 2014.

Cozad, A. and Sahinidis, N. V. A global minlp approach to symbolic regression. *Math. Program., Ser. B*, 170: 97–119, 2018.

Davidson, J. W., Savic, D. A., and Walters, G. A. Symbolic and numerical regression: experiments and applications. *Information Sciences*, 150(1):95–117, 2003.

Gunawan, O., Horesh, L., Nannicini, G., and Zhou, W. Symbolic regression embedding dimensionality analysis, 2017. U. S. Patent Application 20190188256A1, application no. 15/843,118, filed Dec 15, 2017.

Horesh, L., Liberti, L., and Avron, H. Globally optimal mixed integer non-linear programming (MINLP) formulation for symbolic regression. Technical Report 219095, IBM, 2016.

Izzo, D., Biscani, F., and Mereta, A. Differentiable genetic programming. In et al., J. M. (ed.), *EuroGP 2017, LNCS 10196*, pp. 35—51, 2017.

Korns, M. F. Accuracy in symbolic regression. In Riolo, R., Vladislavleva, E., and Moore, J. H. (eds.), *Genetic Programming Theory and Practice IX*, pp. 129—151. Springer, Berlin, 2011.

Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, 1992.

Langley, P. Data-driven discovery of physical laws. *Cognitive Science*, 5(1):31–54, 1981.

Langmuir, I. The adsorption of gases on plane surfaces of glass, mica and platinum. *Journal of the American Chemical Society*, 40(9):1361–1403, 1918. doi: 10.1021/ja02242a004.

Luo, C., Chen, C., and Jiang, Z. A divide and conquer method for symbolic regression. *arXiv:1705.08061v2*, 2017.

Maher, S. J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R. L., Hendel, G., Koch, T., Lübbecke, M. E., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J. T., and Witzig, J. The SCIP optimization suite 4.0. Technical Report 17-12, ZIB, Takustr.7, 14195 Berlin, 2017.

Schmidt, M. and Lipson, H. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

Schmidt, M. and Lipson, H. Symbolic regression of implicit equations. In *Genetic Programming Theory and Practice VII*, pp. 73–85. Springer, Berlin, 2010.

Schmidt, M. and Lipson, H. *Eureqa (Version 0.98 beta) [Software]*. 2014. Available from www.nutonian.com.

Stephens, T. Genetic programming in python, with a scikit-learn inspired api: gplearn. version 0.41, 2019. URL https://gplearn.readthedocs.io/en/stable/.

Tawarmalani, M. and Sahinidis, N. V. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

Udrescu, S.-M. and Tegmark, M. AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.* 6: eaay2631, 2020.

Veeramachaneni, K., Arnaldo, I., Derby, O., and O'Reilly, U.-M. Flexgp: Cloud-based ensemble learning with genetic programming for large regression problems. *Journal Of Grid Computing*, 13(3):391–407, 2015.

Willis, M. J., Hiden, H. G., Marenbach, P., McKay, B., and Montague, G. A. Genetic programming: An introduction and survey of applications. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1997. GALESIA 97. Second International Conference On (Conf. Publ. No. 446)*, pp. 314–319. IET, 1997.