The Metric is the Message: Benchmarking Challenges for Neural Symbolic Regression

Amanda Bertschinger(✉), Q. Tyrell Davis, James Bagrow, and Joshua Bongard

University of Vermont, Burlington, VT 05405, USA
ambertsc@uvm.edu

Abstract. The neural symbolic regression (NSR) literature has thus far been hindered by an over-reliance on individual, *ad hoc* evaluation metrics, producing seemingly favorable performance for the method using it but making comparison between methods difficult. Here we compare the performance of several NSR methods using diverse metrics reported in the literature, and some of our own devising. We show that reliance on a single metric can hide an NSR method's shortcomings, causing performance rankings between methods to change as the evaluation metric changes. We further show that metrics which consider the structure of equations generated after training can help reveal these shortcomings, and suggest ways to correct for them. Given our results, we suggest best practices on what metrics to use to best advance this new field.

Keywords: Benchmarks · Interpretability and explainability · Symbolic regression · Deep learning

1 Introduction

Classical regression encompasses a range of methods that fit an equation to given data. In classical regression, the data and an equation skeleton is provided, to which coefficients are then fit. The limitation of this approach is the need to formulate an appropriate equation, which requires manual analysis of the data to determine an appropriate formal representation. One solution to this issue is symbolic regression (SR) [14]. SR searches over the space of possible equations to explain a given data set, removing the need for manual formulation of the equation and thus avoiding human bias. Traditionally, SR is conducted with genetic programming (GP) [2,12–14,26,27,35]. Advances in GPSR were hard to discern due to a lack of common metrics and benchmark equations. Throughout its history since the 19901990ss, a variety of metrics and best practices for GPSR have been studied [1,16,17]. However, it was not until 2012 that the lack of standardization began to be discussed and benchmarks proposed [15,19,36]. This delay in recognizing the problem a lack of standardized metrics and benchmarks creates hindered the progress of GPSR, and to date there still exist GPSR methods that rely on non-standard metrics and toy datasets that are not a part of established benchmarks.

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14172, pp. 161–177, 2023.
https://doi.org/10.1007/978-3-031-43421-1_10

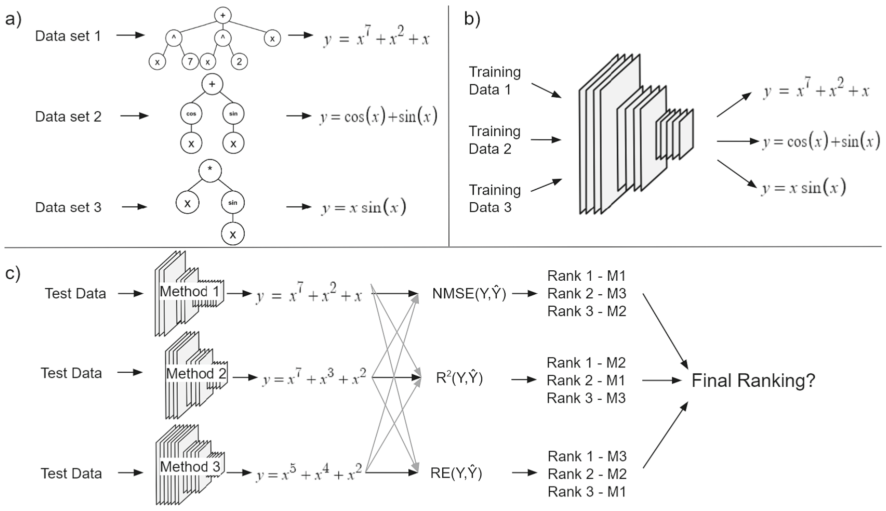

Fig. 1. a) Genetic Programming Symbolic Regression. Each instance is trained on a dataset of (x,y) values to describe a separate equation. b) Neural Symbolic Regression. A network is trained on multiple datasets of (x,y) values that describe a variety of equations. c) NSR methods' use of different metrics obscures how well networks perform in comparison to each other, so a final ranking of the best performing method is unclear.

Regardless of the speed of progress in evolutionary algorithm-based SR, it generally remains the standard for SR. However, such gradient-free search approaches require considerable computation, and they only fit one set of equations to one data set, rather than learning the more general problem of how to transform different data sets into different equations.

Deep learning methods offer a potential approach for achieving this generality. AI models such as large language models [7,25,34] show promise for this generalized form of SR by transforming data into equations [3,4,11,32,33]. This is possible because an equation can be considered a statement composed in the language of mathematical operators and variables. Indeed, recent literature shows progress in using transformers [4,11,32,33], a type of large language model, to generate an equation when fed a given data set.

Currently, however, NSR studies use *ad hoc* metrics on non-standardized benchmark sets, making comparison across methods difficult (Fig. 1c). This is similar to the issues faced by GPSR, which we seek to preemptively avoid. Similar studies have been done in other fields of machine learning in order to help ensure trustworthy benchmarks and comparisons between methods [5]. In order to better compare across methods, here we compare NSR methods (Table 1) using some metrics drawn from the literature and some created by us, and find that this reveals limitations in these current methods that are hidden when only one metric is used. Given our results, we suggest best practices on what benchmark equations and metrics to use to best advance this new field.

2 Methods

Five NSR methods have been reported in the literature to date [3,4,11,32,33]. Each method trains some kind of differentiable network to transform multiple data sets into multiple equations such that, post training, it can transform unseen data sets into equations accurately reflecting that data. This requires a set of training equations and a set of testing equations, and data generated from both. As GPSR is specialized per equation instead of being generalized for any equation, comparison between neural and GPSR methods is unfair given GPSR will always be more accurate than NSR. As such, evolutionary methods are not considered in this paper.

First, we define post generation coefficient fitting (PEGCF) and the role it plays in NSR. Second, benchmark equations in NSR are discussed. Third, the variety of metrics implimented are defined and analyzed. Fourth, details of the NSR methods and are outlined. Finally, we define the control equations that are used for comparison against NSR generated equations.

2.1 Post Equation Generation Coefficient Fitting

It has been found that training a transformer to generate coefficients and equation structure simultaneously is difficult, so some of these models [4,32] are only trained to generate equation structure, and rely on coefficient fitting methods [6,8,10,28] post inference. Even the NSR methods that do predict coefficients during training [11,33] still use PEGCF during and after training to further refine the coefficients in generated equations.

In theory, a trained model could learn to ignore data and instead always generate the same, sufficiently long equation that could be fit to various new data sets by PEGCF. Thus, as an initial demonstration of the utility of employing multiple NSR metrics, we reran some of the NSR methods reported in the literature [4,32,33] but denied them access to PEGCF. Target equations were generated with coefficients of 1, and coefficients in generated equations were replaced with 1; this allows NSR methods to not have to guess or fit coefficients at all. Numeric metrics reveal that several of these methods are overly reliant on PEGCF as the PEGCF-denied methods suffered deteriorations against these metrics; a symbolic metric—equation length—reveals how this over reliance might be manifesting (Sect. 3.1).

2.2 Benchmarks

In NSR there is a lack of commonly-agreed upon benchmark equations. Though some methods [4,33] test on benchmark equations such as the Nguyen equations [31], on their own, these equations tend to lack diversity and complexity. Thus, we propose a set of additional benchmark equations with increasing diversity and complexity (Appendix A).

2.3 Metrics

Choosing appropriate metrics for NSR is complicated by the fact that, unlike most forms of supervised learning, SR investigators face a dilemma between using numeric or symbolic metrics. The former rewards for prediction accuracy while the latter rewards for structural accuracy.

Numeric Metrics. Numeric evaluation metrics measure, in different ways, the distance between the dependent variable values of a given data set (Y) and dependent variable values predicted by the equation generated by a model ($\hat{Y}$) when provided with independent variable values X. Use of such metrics implies the authors are interested in the predictive power of generated equations, rather than their form.

Biggio [4] uses numpy's `isclose` method. Here, we use `isclose` with the specifications reported in [4]: an absolute tolerance of 0.001, relative tolerance of 0.05, and a threshold of 0.95. This means that a generated equation that differs from the target equation at less than 5% of points is considered accurate. This metric is reported as the percentage of equations that are accurate. This metric has the advantage of being able to deal with values of $\pm\infty$ and `NaN`, however the multiple layers of error allowance do slightly obscure the model's true performance.

We also implement Valipour's [32] "protected" normalized mean square error (NMSE)—a different numeric metric—given by

$$\mathrm{NMSE}(Y, \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \hat{y_i})^2}{\|y_i + \epsilon\|_2}, \tag{1}$$

where $\epsilon = 10^{-5}$ [32] is a small factor used to avoid divide by zero errors. Mean squared error is a common metric in general, and using normalization keeps any extreme values from contributing overly much to the individual error. One disadvantage to this metric is that it ranges over $[0, +\infty]$. This can cause large outliers to have an outsized effect on the final mean NMSE, as it can take many perfect scores of 0 to balance one very large score.

Finally, we employ several versions of the coefficient of determination, R^2 [9]. Interpreting raw R^2 can be challenging, as outliers can have a significant effect on the final average. There are various ways to compensate for this. First is the mean raw R^2 score, as used in Vastl [33], given by

$$R^2(Y, \hat{Y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}. \tag{2}$$

This shares a similar disadvantage to NMSE: perfect accuracy results in a value of 1.0, but poor accuracy can incur infinitely negative values. To alleviate the outsized effect outliers can have, median R^2 can be used. Alternately to median, R^2 over a threshold has been reported in the literature as an accuracy metric, but there is no consensus on what threshold to use: Biggio [4] used a threshold of 0.95, Kamienny [11] used 0.99, and La Cava [15] used 0.999.

Symbolic Metrics. Another version of accuracy is to consider the structural distance between a generated equation and a known target equation. Usage of symbolic metrics implies the authors wish to create a method that can 'explain' a data set, in mathematical form, to a human reader. To compare two equations symbolically, they must first be simplified, as NSR models can generate equations with duplicate terms during inference. Due to how symbolic comparison of two equations works, it is necessary to group matching terms and ensure that the equations are always written in the same order (e.g. a polynomial is always written highest to lowest order first). To do this we employ sympy's `simplify` method. This simplifies an equation and allows it to be accessed in string or tree form. This is important as some symbolic metrics require conversion of a string into a tokenized representation in order to calculate error, while other methods require the tree form of an equation.

Cross-entropy is commonly used for symbolic loss during training, and can be similarly used as an evaluation metric for equations generated by a trained model. To use cross-entropy, equations are transformed into a tokenized representation, and then processed using pytorch's `CrossEntropyLoss`.

La Cava [15] and Petersen [24] used another symbolic metric: exact symbolic equivalence. For this metric, first, target and predicted equations are compared in their tree representations $(f(x), \hat{f}(x))$ to determine whether they are the same tree: `simplify(f - f_hat) == 0`. This metric then reports the proportion of exact matches across the training equations or testing equations. La Cava [15] used a slightly more lenient definition of symbolic equivalence that includes equations that differ by a constant or are proportional. It is worth noting that [15,24] report evolutionary SR methods; exact equivalence has not yet been used as a metric in NSR. For the sake of parity with other symbolic metrics (the lower the metric the better the method is performing), we here report symbolic inequivalence- the proportion of equations that do not exactly match across training or testing equations.

The unforgiving nature of exact symbolic equivalence can foil comparisons between partially-working NSR methods because they all achieve a score of zero. So, the final symbolic metric we consider is tree edit distance (TED), a symbolic metric not yet used by any NSR methods. TED algorithms [20–23,37] calculate how many edits it would take to transform one equation's tree representation into another's. Ordered and unordered TED algorithms are possible. In an ordered algorithm, the order in which the nodes appear in the tree matters, so if a term appears in both equations, but different places in the tree, this is considered a necessary edit. Unordered TED does not make this distinction: so long as the term is present in both trees it is correct and does not require an edit. Here, we use APTED [23], an ordered TED. We report this TED normalized by the number of nodes in the target equation tree because generated equations will require, on average, more edits to transform them into longer target equations.

2.4 NSR Methods

We now summarize the methods considered in this study. Biggio 2021 [4], Valipour 2021 [32], and Vastl 2022 [33] are all implemented. Biggio 2020 [3] and Kamienny 2022 [11] were not implemented in this study due to a lack of models provided by the authors.

Biggio 2020. [3] implement a fully-convolutional model with gated recurrent units, residual connections and attention. The model uses cross-entropy loss for training and is evaluated on root mean squared error. It is trained on equations generated by their own equation generation program, and is tested on a combination of equations seen during training, and new equations, still produced by their equation generator. Despite significant effort we were unable to reimplement this method, so it is not included in the cross-method comparison below.

Biggio 2021. [4] use a set transformer encoder with a standard transformer decoder model, trained on cross-entropy loss using millions of equations from their equation generator. The trained model is evaluated by their own numerical metric using `np.isclose` (detailed below) on a combination of the AI Feynman equations [30], Nguyen equations [31], and their own generator-produced equations unseen during training. They provided a pretrained model to the community, which we use below.

Valipour 2021. [32] use a two transformer model, first passing data through a transformer to produce an order-invariant representation then used to train a second transformer that generates equations. The entire model is trained using cross-entropy loss and tested on NMSE. They use their own equation generator to generate training and testing equations. For our testing, we trained their model on equations generated by their generator.

Kamienny 2022. [11] use a fully connected feedforward network to reduce data dimensionality and then feed the lower dimensional data to the transformer model to generate equations from them. Their model predicts equations and coefficients, the latter of which are then refined using a coefficient fitter. The model is trained on cross-entropy loss and tested on both the coefficient of determination (R^2) and accuracy to tolerance.

They use their own equation generator for generating training and part of their testing equations (the rest come from the Feynman [30] and ODE-Strogatz [29] equations). They do not provide their model (code or pretrained), and as such we were unable to implement their method.

Vastl 2022. [33] use a transformer architecture with a cross-attention encoder and self-attention decoder. Their model uses cross-entropy and mean squared error losses for equation structure and coefficients respectively during training, and is tested on both R^2 and relative error. They generate their own training equations, and their testing equations are a combination of their training equations and benchmark equations drawn from the SR literature. They provide a pretrained model, which we use below.

In addition, we considered but did not ultimately test the NSR methods by Biggio [3] and Kamienny [11] as we were unable to implement either method and thus do not report results from them.

2.5 Control Equations

If PEGCF can fit randomly or systematically generated equations to a data set with similar success as it does for an NSR-generated equation, this would indicate that NSR method is not generating structurally correct equations. Thus, we compared NSR-generated equations against three types of control equations: polynomials, Fourier series, and randomly-generated equations. All equations were designed with 10°C of freedom in order to allow sufficient ability to fit to data. The polynomials are

$$f(x) = \sum_{i=0}^{10} c_i x^i, \tag{3}$$

where c_i are the fit coefficients. The Fourier series used was

$$f(x) = c_0 + \sum_{i=1}^{5} c_{i,1} * \sin(ix + c_{i,2}). \tag{4}$$

Lastly, the random equations are generated using Valipour's [32] tree based equation generation with access to the mathematical operations [+, $*$, \, sin, cos, $\sqrt{}$, exp, log] and a maximum tree depth of 9.

3 Results

We use a variety of numeric and symbolic metrics to analyze the predictive and structural accuracies of equations generated by various methods. Methods that provide pre-trained networks use those saved weights for evaluation. Other methods that do not provide such are trained from scratch using the network architecture provided.[1] All methods' networks are trained on their own generated equations and are evaluated on our benchmark equations (Appendix A).

3.1 Numeric Metrics

Cross-Method Comparison. Results obtained against the numeric metrics are shown in columns 2–4 of Table 1. Biggio [4], with the `isclose` metric, outperforms Vastl [33] with this metric ($p \ll 0.001$). No significant performance advantage over Valipour [32] was detected ($p > 0.05$) with this metric. Similarly, although it appears that Valipour [32] performs best using their NMSE metric, there is no significant difference compared to other NSR methods using NMSE. Overall, Table 1 shows that the relative ranking of NSR methods is dependent ~~on the evalua~~tion metric used.

[1] https://github.com/mec-lab/metric_message.

Table 1. Comparison of testing metrics for NSR methods. Bold values indicate that it was the metric used in that method. Bracketed values denote the 95% confidence interval for the metric. Mann-Whitney U [18] tests for significance were performed within each column. Bonferroni correction was employed to compensate for the 12 pairwise comparisons. In the case of NMSE and R^2, the median is reported instead of the mean due to unbounded ranges.

Reference	NMSE	$R^2 > 0.99$	R^2	`np.isclose`	Cross entropy	Symb Equiv	Tree Edit Dist.Prop
Biggio et al. (2021)	0.20 (0.16, 0.25)	0.34 (0.33, 0.36)	0.95 (0.93, 0.96)	**0.24** *** (0.23, 0.26)	0.60 (0.60, 0.60)	0.18 (0.17, 0.19)	1.01 (0.99, 1.02)
Valipour et al. (2021)	**0.03** (0.003, 0.07)	0.37 (0.19, 0.59)	0.91 (0.29, 0.99)	0.07 (0.01, 0.31)	1.96 (1.56, 2.30)	0.07 (0.00, 0.30)	1.05 (0.83, 1.18)
Vastl et al. (2022)	0.72 (0.64, 0.84)	0.15 (0.31, 0.47)	**0.38** (0.18, 0.20)	0.13 (0.12, 0.15)	0.48 (0.46, 0.49)	0.05 (0.05, 0.07)	1.69 (1.64, 1.75)

With and Without PEGCF. Here we compare NSR methods with and without PEGCF, providing insight into how much data-to-equation effort is performed by skeleton generation (NSR) and coefficient fitting (PEGCF). We found that, when denied access to PEGCF, Biggio [4] and Valipour [32] suffered significant deterioration according to two numerical metrics during testing. This is similar to Kamienny [11], which reports a better R^2 value when PEGCF is used compared to when it is not. Vastl [33] however does not suffer any statistically significant deterioration according to either metric (Table 2).

Equation Length Comparison. One hypothesis for how NSR methods come to overly rely on PEGCF is that the models learn to generate equations that have too many terms and rely on PEGCF to lessen the contribution of these extra terms. We can check whether this occurs by examining the lengths of generated equations compared to our benchmark targets. Table 3 shows comparisons between generated and target equation length. In all methods, there is a significant difference between the two. Valipour's method [32] tends to generate overly-long equations while Biggio [4] and Vastl [33] tend to generate overly-short equations. Figure 2 shows the distribution of coefficient magnitudes across all methods, showing that while the correct coefficient of 1.0 is frequently predicted, there are many occurrences of the coefficients predicted being close to zero or very large.

Table 2. Median results over all benchmark equations for two different numeric evaluations metrics, with and without optimization techniques. Median is used for all metrics due to unbounded ranges. The bracketed pairs of values denote the 95% confidence interval for each median. *** denotes $p \ll 0.001$ after the Bonferroni correction is applied to compensate for the six pairwise comparisons.

	NMSE		R^2	
	PEGCF	no PEGCF	PEGCF	no PEGCF
Biggio et al (2021)	0.20*** (0.16, 0.25)	0.43 (0.38, 0.47)	0.95*** (0.93, 0.96)	0.47 (0.39, 0.54)
Valipour et al (2021)	0.03*** (0.003, 0.07)	0.52 (0.15, 1.45)	0.91*** (0.29, 0.99)	-0.12 (-1.03, 0.51)
Vastl et al (2022)	0.72 (0.64, 0.84)	0.77 (0.69, 0.91)	0.38 (0.31, 0.47)	0.33 (0.26, 0.42)

Table 3. Comparison of equation length for equations predicted by networks. *** denotes $p \ll 0.001$ after the Bonferroni correction is applied to compensate for the three pairwise comparisons.

Reference	Target		Predicted
Biggio	10.625 (10.51, 10.75)	***	7.45 (7.36, 7.54)
Valipour	12.74 (12.69, 12.78)	***	17.92 (17.87, 17.99)
Vastl	11.11 (10.94, 11.26)	***	9.69 (9.41, 10.02)

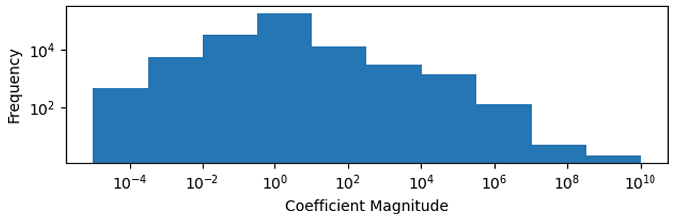

Fig. 2. Distribution of PEGCF coefficients across all equations generated by implimented NSR methods. The horizontal axis is the magnitude of the coefficients $|c_i|$. The vertical axis reports the number of appearances of that coefficient magnitude across all equations.

Comparison to Control Equations. The reliance of NSR methods on PEGCF can be so great that, in some cases, NSR-generated equations require no less coefficient fitting effort than is required to fit random equations, Fourier series, or polynonimals with near-equivalent degrees of freedom (Table 4). With enough terms, these control equations become competitive with NSR-generated networks, if numeric metrics are used to the compare them. Here we take competitive to mean that the reported median R^2 score is within or above the confidence interval of the NSR models for that benchmark equation.

Table 4. Performance of NSR methods and control equations on R^2 with PEGCF for benchmark equations [31]. Highlighted are cases where the control methods are competitive with the NSR methods, where competitive is defined as the control median above or equal to the best performing NSR method median. Median is used for all cases due to unbounded ranges in R^2.

		Random	Fourier	Polynomials	Valipour	Biggio	Vastl
Nguyen							
1	$x^3 + x^2 + x$	−0.03	−1.22	**1.0**	1.0	1.0	1.0
2	$x^4 + x^3 + x^2 + x$	0.54	−0.09	**1.0**	1.0	0.99	1.0
3	$x^5 + x^4 + x^3 + x^2 + x$	0.36	−0.21	**1.0**	1.0	1.0	0.89
4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	0.16	−0.14	**1.0**	1.0	0.90	0.80
5	$\sin x^2 \cos x - 1$	0.08	-22.62	**0.71**	0.51	−3.08	−0.5
6	$\sin x + \sin x + x^2$	0.40	−1.67	**0.48**	0.47	0.06	0.07
7	$\log x + 1 + \log x^2 + 1$	−0.41	**1.0**	−2.7E+5	1.0	0.98	0.86
8	$\sqrt{x}$	0.27	–	0.03	1.0	1.0	-8.92
Complexity spanning							
1	$\sin(x * e^x)$	0.11	−0.57	0.27	0.14	1.0	−0.30
2	$x + \log(x^4)$	-0.19	0.86	−7.0E+4	0.98	1.0	1.0
3	$1 + x \sin(\frac{1}{x})$	0.13	0.76	−0.06	0.96	0.63	0.82
4	$\sqrt{x^3} \log(x^2)$	**−0.40**	**−0.09**	−1.3E+5	−0.57	–	–
5	$\frac{x}{\sqrt{x^2+\sin x}}$	−0.01	−9.63	0.02	−0.01	−0.33	−0.08
6	$\frac{x+x^3}{1+x\cos x^2}$	0.91	0.85	**0.99**	0.92	−1.24	−0.94
7	$\frac{e^x(1+\sqrt{1+x}+\cos x^2)}{x^2}$	−13.10	−9.62	−3.2E+6	0.58	0.07	0.26
8	$\cos\left(\frac{x+\sin x}{x^3+x\log x^2}\right)$	−0.03	0.00	−0.03	0.0	–	–

3.2 Symbolic Metrics

Unlike numeric metrics, Table 5 shows that all NSR methods statistically significantly outperform all control equations for all symbolic metrics apart from Valipour [32], which does not outperform random equations when measured with cross-entropy loss.

4 Discussion

NSR models are not performing as well as the statistics reported in their literature would seem to imply. First, varied metrics make cross-method comparison difficult. Second, PEGCF artificially makes NSR models' performance seem better. Third, we show that use of symbolic metrics can help reveal the true performance of the models though the metric must be carefully chosen. Finally, we show why it is important to consider both numeric and symbolic metrics.

The relative ranking of methods is fluid. Because of the change in relative performances when switching metrics, it is near impossible to compare models

Table 5. A comparison of symbolic metrics on control methods and NSR methods for a) cross-entropy loss, b) symbolic inequivalence, and c) tree edit distance proportion, showing the lower values for across all metrics for NSR methods. The bracketed pairs of values denote the 95% confidence interval for each median. *** denotes $p \ll 0.001$ for the NSR model after the Bonferroni correction is applied to compensate for the 27 pairwise comparisons. In all but one case, the pairwise comparison shows the NSR method out-competing the control methods.

a)

	Biggio [3] 0.6 (0.6,0.6)	Valipour [27] 1.96 (1.56, 2.30)	Vastl [28] 0.48 (0.46, 0.49)
Random 1.8 (1.75, 1.85)	***	p¿0.05	***
Fourier 3.48 (3.48,3.48)	***	***	***
Polynomial 3.67 (3.61,3.72)	***	***	***

b)

	Biggio [3] 98.99 (98.98, 99.01)	Valipour [27] 98.95 (98.82, 99.17)	Vastl [28] 98.31 (98.25, 98.36)
Random 100 (100, 100)	***	***	***
Fourier 100 (100, 100)	***	***	***
Polynomial 100 (100, 100)	***	***	***

c)

	Biggio [3] 0.18 (0.17, 0.19)	Valipour [27] 0.07 (0.0, 0.3)	Vastl [28] 0.05 (0.05, 0.07)
Random 2.83 (2.51, 3.16)	***	***	***
Fourier 14.76 (14.52, 15.0)	***	***	***
Polynomial 3.26 (2.69, 1.02)	***	***	***

cross-method. These metrics, when used on their own, are also problematic in that arbitrary hyperparameters must be chosen (as in the case of $R^2 > 0.99$ and $R^2 > 0.95$) or are hidden within software libraries (as in the case of `np.isclose`).

The decrease in predictive performance of most NSR methods when PEGCF is not allowed (as indicated by the numeric metrics) indicates that the methods are not generating equations with appropriate skeletons. We hypothesize that PEGCF may be performing the bulk of fitting here because NSR learns to either generate very long equations and then PEGCF removes extra terms by setting the coefficients on said terms sufficiently small that they contribute negligibly to numeric metrics, or generate only the part of an equation that contributes most to the data and the PEGCF emphasizes it with sufficiently large coefficients. To test this assertion, we measured the difference in mean number of nodes in the tree representation of the target and NSR generated equations. Shown in

Table 3, the significant disparity between these lengths for all methods shows that the trained models do not generate the structurally correct equations. Instead, good performance of the models is due to PEGCF being able to fit structurally incorrect equations well to the provided data. Overly long equations imply a relationship that is not present in data, while equations missing terms fail to report some relationships. Seen in Table 3, [32] suffers from errors of commission while [4,33] suffers from errors of omission. Across all methods, Fig. 2 shows that PEGCF is often "removing" terms from predicted equations by setting the coefficients for such terms close to zero, and emphasizing contribution from the correctly predicted terms by setting the coefficients for these to be much larger than 1. For those methods where there is no significant effect of PEGCF denial, the networks predict the starting coefficients when generating equations. These coefficients are then further refined by use of PEGCF after training. Presumably, the insignificant change observed in both methods are due to the effects of this during-training coefficient prediction reducing the effects of PEGCF withdrawal.

Symbolic metrics are important in showing how accurately an NSR method is performing on its primary task: automating the distillation of data into equations. Unlike numeric metrics, symbolic metrics are not influenced by PEGCF as they directly compare the symbolic structure of the equation. Of the symbolic metrics, TED is most meaningful as it has the advantages of absolute equivalence, as well as the ability to reveal a gradient in how "close" two equations are. A TED of 0 shows absolute equivalence just as symbolic equivalence does, but TED is not restricted to binary outcomes and could thus show that a particular change to an NSR method improves or harms it.

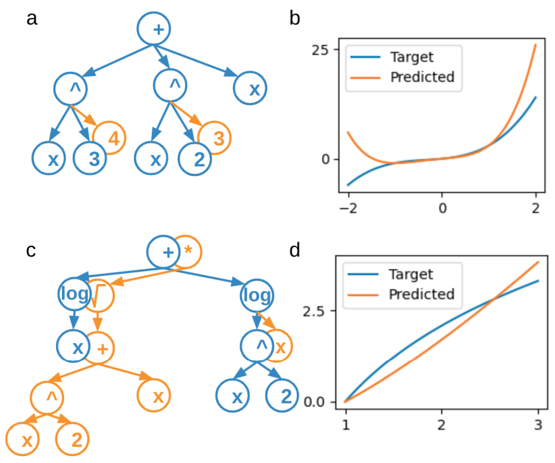

Fig. 3. a) Target versus predicted equation trees for the equations $x^3 + x^2 + x$ and $x^4 + x^3 + x$ respectively. b) Graph of aforementioned equations. c) Target versus predicted equation trees for the equations $\log(x) + \log(x^2)$ and $(\sqrt{x^2 + x})\log(x)$ respectively. d) Graph of these equations.

However, only using symbolic metrics is insufficient to determine how well a network is performing. While symbolic metrics do hold advantages over numeric metrics, the opposite is also true. Foremost is that numeric evaluation metrics do not require a ground truth equation while symbolic ones do. In practice, after training, most data sets will not have a ground truth equation. In such a situation only numeric metrics can be used.

Additionally, there are situations in which two equations can be close symbolically but distant numerically. Figure 3a,b shows two such equations. The TED between $x^3 + x^2 + x$ and $x^4 + x^3 + x$ is 2 (Fig. 3a). However, Fig. 3b shows that the two are quite far numerically, particularly in the negative x range. In a case like this, it would be particularly important to also consider a numeric metric, as that would reveal where the network's failings are more than the symbolic metric. Figure 3c,d show the opposite case: the TED score is large (Fig. 3c), but within the tested range, the equations are numerically close (Fig. 3d). In either case, looking at only the symbolic or numeric metric would be deceptive.

5 Conclusion

NSR is a relatively new field, and like early ML research, it is suffering not just from a lack of commonly-agreed upon metrics, but an assumption that a single metric will suffice. Based on the results we obtained, it is important to report both numeric and symbolic metric scores. We recommend use of median R^2 as a numeric metric and TED (tree edit distance) proportion as a symbolic metric. R^2 is a commonly used, informative, and understandable metric by which to tell how close predictions made by generated equations are to target data. The weakness of R^2 to outliers is mitigated by use of the median instead of the mean. TED proportion is similarly informative and understandable in revealing a generated equation's structural proximity to a target equation. Together, the two metrics give a comprehensive view of a NSR model's accuracy.

Given this, multi-objective optimization for training and testing should be pursued. Common multi-objective optimization methods may help balance the importance of numeric and symbolic metrics depending on the goal of the specific model: some methods may wish to emphasize prediction while others may wish to reveal discovered relationships. Multi-objective optimization can go beyond just evaluation metrics; it may allow for training the models against all of the objectives by using a multi-objective loss function.

We also recommend the NSR field use common training sets. Cross-method comparison may be possible with standardized metrics, but will still be obstructed by training and testing on different equations. As such, we recommend the use of the Nguyen equations [31] as well as our benchmark equations (Table 4) which are designed to offer a diversity of mathematical operations and incrementally increasing complexity.

The history of machine learning demonstrates that no one metric will ever perfectly capture the goals of a community. Diminishing returns against technical metrics like accuracy, over time, revealed to the ML community that other metrics like time, compute, and carbon are also relevant. For the NSR field, acknowledging the importance of numeric and symbolic metrics, as well as standardization of evaluation equations, is the first step. An early acceptance of multi-objective metrics and methods could allow the field to accelerate and prepare for as-of-yet undiscovered metrics of import.

Acknowledgment. This research is based upon work supported by the Broad Agency Announcement Program and Cold Regions Research and Engineering Laboratory (ERDC CRREL) under Contract No. W913E521C0003. Computations were performed on the Vermont Advanced Computing Center supported in part by NSF award No. OAC-1827314. QTD was supported by the NSF under the EFRI program (EFMA-1830870).

A Appendix

See Table 6.

We use a description length definition of equation complexity: the length of a node traversal for the expression tree representation after applying SymPy's `simplify` function to the equation. Shannon's diversity operates on the same

Table 6. Complexities, Shannon's Diversities for Nguyen benchmark [31] and equations designed for this paper to have a steady ramp in complexity and diversity.

Number	Equation	Complexity	Shannon's Diversity
Nguyen-1	$x^3 + x^2 + x$	8	1.93
Nguyen-2	$x^4 + x^3 + x^2 + x$	11	2.17
Nguyen-3	$x^5 x^4 + x^3 + x^2 + x$	14	2.34
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	17	2.48
Nguyen-5	$\sin x^2 \cos x - 1$	9	2.15
Nguyen-6	$\sin x + \sin x + x^2$	9	2.05
Nguyen-7	$\log x + 1 + \log x^2 + 1$	11	2.34
Nguyen-8	$\sqrt{x}$	3	1.10
C.S.-1	$sin(x * e^x)$	5	1.61
C.S.-2	$x + \log(x^4)$	6	1.73
C.S.-3	$1 + x \sin(\frac{1}{x})$	8	2.03
C.S.-4	$\sqrt{x^3} \log(x^2)$	10	2.25
C.S.-5	$\frac{x}{\sqrt{x^2+\sin x}}$	10	2.15
C.S.-6	$\frac{x+x^3}{1+x \cos x^2}$	14	2.40
C.S.-7	$\frac{e^x(1+\sqrt{1+x}+\cos x^2)}{x^2}$	17	2.71
C.S.-8	$\cos\left(\frac{x+\sin x}{x^3+x \log x^2}\right)$	19	2.76

tree representation nodes, and is defined as

$$H = -\sum_{i=1}^{n} p_i \ln(p_i) \tag{5}$$

where p_i is the relative incidence of the *ith* node label.

B Ethics

Symbolic regression has far reaching applications, as any field that requires the collection of data that then needs to be understood in a compact format can benefit from the automatic distillation of said data into an equation. It is the belief of the authors that the ethics of said applications is an important subject to consider. As an application of AI that uses neural networks, it suffers from the general ethics concerns that AI does: bias in both data and perception of results, as well as potential for discriminatory practices.

A key ethical issue in all AI is bias and discrimination. Like other AI networks, symbolic regression networks could easily perpetuate biases and discriminatory practices if trained on biased or incomplete datasets. As an equation that describes a biased dataset would inherently be biased, SR networks could produce equations that accurately describe biased data without it being clear that there is a bias at all. Equations are almost never the true end goal of analyzing data. Instead, said equations are used to further understand the data it describes. The broad applications of neural SR are a key way that potential discriminatory data and practices could be introduced. Potential underlying biases in data could lead to networks that produce equations that are used to perpetuate discriminatory practices.

The ethical concerns that are more unique to SR are generally centered around the people who will be using the network and its resulting equations. There is a cognitive bias toward authority in people, and generally this bias is toward a trust in perceived authority. Through schooling, we have been trained to trust math as an authority, and as SR produces equations- which are math- users will be biased toward trusting the equation without verifying accuracy. Through this, SR can create an illusion of authority where there may not be one. Another user bias that must be considered is the accessibility of SR networks. By nature, an SR network is geared toward the mathematically literate, which presents an exclusionary bias toward people who are not.

Misunderstanding NSR methods and results, such as underestimating the effects of PEGCF discussed in this paper, can exacerbate these problems. Misunderstanding the potential flaws in the validity of the results contributes toward users finding authority in the equations produced regardless of the potential lack of any true accuracy. Our paper seeks to bring to light what contributes to NSR performance with explicit comparisons and test. In doing so, it has the potential to help prevent the false authority that NSR produced equations could cause by making users aware of the potential flaws. Overall, it will be critical to continue to address ethical issues and ensure any research in this field is done carefully and with full awareness of necessary precautions around the potential applications.

References

1. Aldeia, G.S.I., de França, F.O.: Interpretability in symbolic regression: a benchmark of explanatory methods using the Feynman data set. Genet. Program Evolvable Mach. **23**(3), 309–349 (2022)
2. Arnaldo, I., Krawiec, K., O'Reilly, U.M.: Multiple regression genetic programming. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 879–886 (2014)
3. Biggio, L., Bendinelli, T., Lucchi, A., Parascandolo, G.: A seq2seq approach to symbolic regression. In: Learning Meets Combinatorial Algorithms at NeurIPS2020 (2020)
4. Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A., Parascandolo, G.: Neural symbolic regression that scales. In: International Conference on Machine Learning, pp. 936–945. PMLR (2021)
5. Bouthillier, X., et al.: Accounting for variance in machine learning benchmarks. CoRR abs/2103.03098 (2021). https://arxiv.org/abs/2103.03098
6. Broyden, C.G.: The convergence of a class of double-rank minimization algorithms 1. General considerations. IMA J. Appl. Math. **6**(1), 76–90 (1970). https://doi.org/10.1093/imamat/6.1.76
7. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch (2011)
8. Fletcher, R.: A new approach to variable metric algorithms. Comput. J. **13**(3), 317–322 (1970). https://doi.org/10.1093/comjnl/13.3.317
9. Glantz, S.A., Slinker, B.K.: Primer of Applied Regression & Analysis of Variance, 3rd edn (2016)
10. Goldfarb, D.: A family of variable-metric methods derived by variational means. Math. Comput. **24**(109), 23–26 (1970). http://www.jstor.org/stable/2004873
11. Kamienny, P.A., d'Ascoli, S., Lample, G., Charton, F.: End-to-end symbolic regression with transformers. arXiv preprint arXiv:2204.10532 (2022)
12. Kommenda, M., Burlacu, B., Kronberger, G., Affenzeller, M.: Parameter identification for symbolic regression using nonlinear least squares. Genet. Program Evolvable Mach. **21**(3), 471–501 (2020)
13. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge (1994)
14. Koza, J., Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. A Bradford Book, Bradford (1992). https://books.google.com/books?id=Bhtxo60BV0EC
15. La Cava, W., et al.: Contemporary symbolic regression methods and their relative performance. In: Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1) (2021)
16. La Cava, W., Spector, L., Danai, K.: Epsilon-lexicase selection for regression. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. GECCO '16, pp. 741–748. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2908812.2908898, https://doi-org.ezproxy.uvm.edu/10.1145/2908812.2908898
17. Liskowski, P., Krawiec, K.: Discovery of search objectives in continuous domains. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '17, pp. 969–976. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071344

18. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. **18**(1), 50–60 (1947)
19. McDermott, J., et al.: Genetic programming needs better benchmarks. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. GECCO '12, pp. 791–798. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2330163.2330273, https://doi-org.ezproxy.uvm.edu/10.1145/2330163.2330273
20. Ouangraoua, A., Ferraro, P.: A constrained edit distance algorithm between semi-ordered trees. Theoret. Comput. Sci. **410**(8), 837–846 (2009). https://doi.org/10.1016/j.tcs.2008.11.022, https://www.sciencedirect.com/science/article/pii/S0304397508008621
21. Pawlik, M., Augsten, N.: RTED: a robust algorithm for the tree edit distance. Proc. VLDB Endow. **5**(4), 334–345 (2011)
22. Pawlik, M., Augsten, N.: Efficient computation of the tree edit distance. ACM Trans. Database Syst. **40**(1), 1–40 (2015)
23. Pawlik, M., Augsten, N.: Tree edit distance: robust and memory-efficient. Inf. Syst. **56**, 157–173 (2016)
24. Petersen, B.K., Larma, M.L., Mundhenk, T.N., Santiago, C.P., Kim, S.K., Kim, J.T.: Deep symbolic regression: recovering mathematical expressions from data via risk-seeking policy gradients. In: International Conference on Learning Representations (2020)
25. Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training (2018)
26. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science (Am. Assoc. Adv. Sci.) **324**(5923), 81–85 (2009)
27. Schmidt, M.D., Lipson, H.: Coevolution of fitness predictors. IEEE Trans. Evol. Comput. **12**(6), 736–749 (2008). https://doi.org/10.1109/TEVC.2008.919006
28. Shanno, D.F.: Conditioning of quasi-Newton methods for function minimization. Math. Comput. **24**(111), 647–656 (1970). http://www.jstor.org/stable/2004840
29. Strogatz, S.H.: Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. CRC Press, Boca Raton (2018)
30. Udrescu, S.M., Tegmark, M.: AI Feynman: a physics-inspired method for symbolic regression. Sci. Adva. **6**(16), eaay2631 (2020)
31. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genet. Program Evolvable Mach. **12**(2), 91–119 (2011)
32. Valipour, M., You, B., Panju, M., Ghodsi, A.: SymbolicGPT: a generative transformer model for symbolic regression. arXiv preprint arXiv:2106.14131 (2021)
33. Vastl, M., Kulhánek, J., Kubalík, J., Derner, E., Babuška, R.: SymFormer: end-to-end symbolic regression using transformer-based architecture. arXiv preprint arXiv:2205.15764 (2022)
34. Vaswani, A., et al.: Attention is all you need (2017)
35. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.: Improving model-based genetic programming for symbolic regression of small expressions. Evol. Comput. **29**(2), 211–237 (2021)
36. White, D.R., et al.: Better GP benchmarks: community survey results and proposals. Genet. Program Evolvable Mach. **14**(1), 3–29 (2013)
37. Zhang, K.: A constrained edit distance between unordered labeled trees. Algorithmica **15**, 205–222 (1996). https://doi.org/10.1007/BF01975866