SymbolicGPT: A Generative Transformer Model for Symbolic Regression

Mojtaba Valipour
David R. Cheriton School of Computer Science
University of Waterloo
mojtaba.valipour@uwaterloo.ca

Bowen You
University of Waterloo
byyou@uwaterloo.ca

Maysum Panju
University of Waterloo
mhpanju@uwaterloo.ca

Ali Ghodsi
Department of Statistics and Actuarial Science
University of Waterloo
ali.ghodsi@uwaterloo.ca

Abstract

Symbolic regression is the task of identifying a mathematical expression that best fits a provided dataset of input and output values. Due to the richness of the space of mathematical expressions, symbolic regression is generally a challenging problem. While conventional approaches based on genetic evolution algorithms have been used for decades, deep learning-based methods are relatively new and an active research area. In this work, we present SymbolicGPT, a novel transformer-based language model for symbolic regression. This model exploits the advantages of probabilistic language models like GPT, including strength in performance, scalability, and flexibility. Through comprehensive experiments, we show that our model performs strongly compared to competing models.[1]

1 Introduction

Deep language models have made an enormous impact in the field of linguistics and natural language processing. With the advances in technology like Generative Pre-trained Transformers (GPT) Radford et al. (2018), the scope of problems now accessible to language models continues to grow. It is particularly interesting when language models are used for tasks that, at first glance, do not seem to have any relationship with language at all.

Symbolic regression, the problem of finding a mathematical equation to fit a set of data, is one such task. The objective of symbolic regression is to obtain a closed-form symbolic mathematical expression to describe the relationship between specified predictor and response variables, where the mathematical expression is allowed to be flexible without being restricted to a particular structure or family. More precisely, the goal in symbolic regression is to recover a mathematical function f in terms of the input variables $\boldsymbol{x} = [x_1 \dots x_d]^\top$, given a set of data point vectors of the form $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, such that $f(\boldsymbol{x}_i) = y_i$ for all i. Here, $x_1, \dots, x_d, y_i$ are scalars and $\boldsymbol{x}_i \in \mathbb{R}^d$.

In this work, we explore a novel approach to symbolic regression by considering it as a task in language modeling. Symbolic mathematics behaves as a language in its own right, with well-formed mathematical expressions treated as valid "sentences" in this language. As with human language, symbolic equations follow their own grammar, and an equation cannot be generated at random. It is natural, therefore, to consider using deep language models to address tasks involving symbolic mathematics.

[1]Code, datasets, and results are publicly available at https://github.com/mojivalipour/symbolicgpt

We can frame the regression problem as an exercise in captioning. Each instance takes input in the form of a cloud of points in $\mathbb{R}^{d+1}$, with each point consisting of d components corresponding to $\boldsymbol{x}$ and a single component for the associated y value. The instance returns a statement in the language of symbolic mathematics to describe the point set. By training a model to correctly "caption" datasets with the equations underlying them, we obtain a system for performing symbolic regression quickly and accurately.

2 Related Work

Traditionally, the problem of symbolic regression has been tackled with methods based on genetic algorithms McKay et al. (1995); Augusto and Barbosa (2000); Schmidt and Lipson (2009); Murari et al. (2014); Wang et al. (2019). This is, however, computationally expensive, highly randomized, requires instance-based training, and struggles with learning equations containing many variables and constants.

More recently, newer approaches to symbolic regression have arisen that make use of neural networks. The EQL (Equation Learner) model Martius and Lampert (2017); Sahoo et al. (2018) and others based on it Chen (2020); Kim et al. (2020), take advantage of advances in deep learning as an alternative to genetic approaches. However, they still approach symbolic regression as an instance-based problem, training a model from scratch given every new input dataset for a regression task.

A recent study Biggio et al. (2020) presents a novel, language-based method for handling symbolic regression as a machine translation task, similar to the approach used by Lample and Charton (2020) for performing symbolic integration and solving differential equations. Although this method overcomes the cost of per-instance training, its interpretation of the input dataset as a textual string limits its usability, as the input data must follow specific constraints, such as fitting a one-dimensional mesh of fixed size. Consequently, this method can only be used in one-dimensional space. However, in most problems, more than one variable is involved and we need to find a multivariate function.

A recent extension of this work Biggio et al. (2021) uses the set transformer encoder Lee et al. (2019) instead of LSTMs to resolve the scalability issues and address the permutation invariancy of the input points properly. In this work, we also propose a scalable method that removes such limitations on the structure of input data.

3 Method

Our model for symbolic regression, SymbolicGPT, consists of three main stages: obtaining an order-invariant embedding of the input dataset using our modified T-net network Qi et al. (2017), obtaining a skeleton equation using a GPT language model Radford et al. (2019), and optimizing constant values to fill in the equation skeleton. In addition to discussing each of these steps, we also present the method for generating our equation datasets.

3.1 Equation Generation

To train our language model, we need a large dataset of solved instances of symbolic regression. This dataset is a collection of input-label pairs where each input is in the form of a numerical dataset, itself a set of input and output pairs $\{(\boldsymbol{x}, y)\}$, and the corresponding label is a string encoding the symbolic expression governing the relationship between variables in the numerical dataset.

For our training dataset, we use an approach similar to Lample and Charton (2020), where we start with a blank parse tree and then "decorate" the nodes with choices of operators and variables. In contrast with Lample and Charton (2020), we do not constrain our parse trees by the number of nodes, but by the number of levels. This enables more control over the maximum level of complexity in the equations used in our training set, as the number of levels in the parse tree corresponds to the number of potential function nesting, a measure of how complex an equation can be.

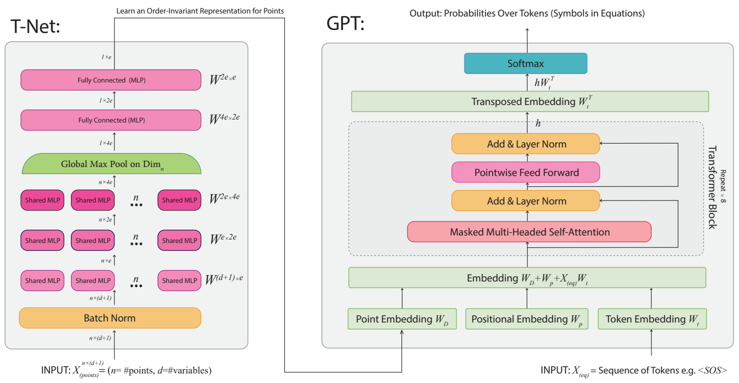
3.2 Order-Invariant Embedding

The first step in our system is to convert the input dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^{d+1}$ into a single vector embedding $\boldsymbol{w}_D \in \mathbb{R}^e$. For the conversion to be useful, it must have two properties. First, it should not strictly depend on the number of points in the input dataset, n. In practice, the datasets provided as input to a symbolic regression solver may have varying sizes, and we do not want our method to be restricted to cases with a fixed number of input points.

Second, the conversion method should not be sensitive to the order in which the points of the dataset are given. The input to a symbolic regression instance is a collection of data points, rather than a sequence, and the optimal symbolic expression to fit the dataset should not depend on the order in which the points are listed. Thus, the vector embedding of the dataset should be similarly order-invariant.

Our approach for converting the dataset D into a vector embedding is to use a network similar to T-net, a kind of deep network that makes use of a global max-pooling layer to provide order-invariance over its arbitrarily-sized input Qi et al. (2017). Our T-net takes as input the dataset D, consisting of n data points over d variables, represented in matrix format as $X \in \mathbb{R}^{n\times(d+1)}$, where n can be any number and d, the number of allowable variables, is fixed in advance. Any symbolic regression instance with fewer than d variables can be padded with 0 values, bringing the total number of variables up to d.

3.3 Generative Model Architecture



T-Net:
Learn an Order-Invariant Representation for Points
$1\times e$
Fully Connected (MLP)
$W^{2e\times e}$
$1\times 2e$
Fully Connected (MLP)
$W^{4e\times 2e}$
$1\times 4e$
Global Max Pool on Dim_n
$n\times 4e$
Shared MLP
n
$W^{2e\times 4e}$
$n\times 2e$
$W^{e\times 2e}$
$n\times e$
$W^{(d+1)\times e}$
$n\times(d+1)$
Batch Norm
INPUT: $X^{n\times(d+1)}_{(points)} = (n = \#\text{points}, d = \#\text{variables})$
GPT:
Output: Probabilities Over Tokens (Symbols in Equations)
Softmax
hW_t^T
Transposed Embedding W_t^T
h
Add & Layer Norm
Pointwise Feed Forward
Masked Multi-Headed Self-Attention
Transformer Block
Repeat × 8
Embedding $W_D+W_p+X_{(eq)}W_t$
Point Embedding W_D
Positional Embedding W_p
Token Embedding W_t
INPUT: $X_{(eq)}$ = Sequence of Tokens e.g. $<SOS>$

Figure 1: The architecture of SymbolicGPT. The left box illustrates the structure of our order-invariant T-net for obtaining a vector representation of the input dataset, and the right box shows the structure of the GPT language model for producing symbolic equation skeletons.

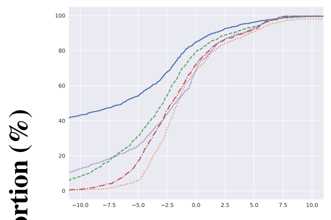
The main component of SymbolicGPT is the deep network for producing symbolic equations, as implemented using a GPT-based language model Radford et al. (2018, 2019); Brown et al. (2020). This framework takes in two pieces of input: the order-invariant embedding of the point cloud $\boldsymbol{w}_D$ as produced by the T-net, representing the input dataset, and a sequence of tokens, $X_{(eq)}$, used to initialize the output formula string.

To train our model, we used the standard unsupervised language model's cost function as mentioned in the original GPT paper Radford et al. (2018).

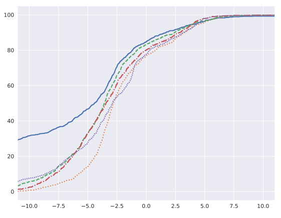
Once the trained GPT model predicts a skeleton equation, we learn values of constants using BFGS optimization Fletcher (1987) to decorate the skeleton as a post-processing step. This division of tasks is a common approach for string-based regression methods Kommenda et al. (2019); Biggio et al. (2020).

4 Experiments and Results

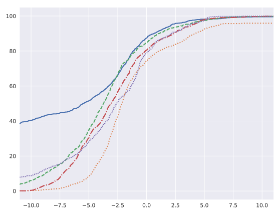
Our experimental framework consists of a large-scale comparison test where we test our model on 1000 different, randomly generated instances of symbolic regression and evaluate performance based on $MSE_N(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i-\hat{y}_i)^2}{\|\boldsymbol{y}+\epsilon\|_2}$. We repeat this test on five different settings, based on the choice of the dimension d: datasets with one input variable, two variables, three variables, and in two different test sets, a random selection between one to nine variables. This last pair of tests will be referred to as the "general" experiments. The first general experiment, along with the three experiments before it, use equations that come from the same distribution that we used to generate



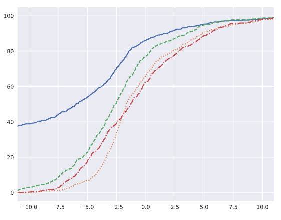
(a) 1 variable



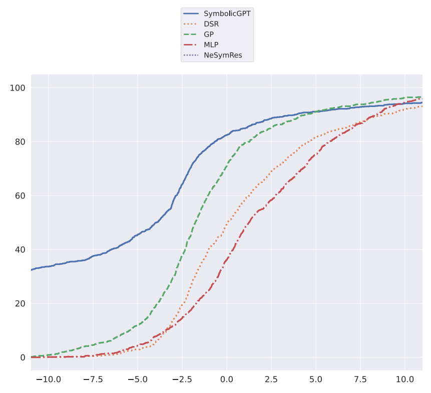
(c) 3 variables



(b) 2 variables



(d) 1 to 9 variables (General)



(e) 1 to 9 variables: This experiment includes all of the benchmark equations in both AI Feynman Udrescu and Tegmark (2020) and Nguyen benchmarks Uy et al. (2011). We used these benchmarks because it is a common practice in the literature.

Log Normalized MSE

Figure 2: Cumulative $\log MSE_N$ over all methods and experiments in a limited budget setting. Each curve shows the proportion of test cases that attained an error score less than every given threshold. SymbolicGPT finds better fitting equations for more test cases than any other method tested, as well as more highly accurate equations (with $\log MSE_N < -10$).

the training set. To ensure that the test set aligns with the training set in terms of constants and data points, and to evaluate the model on real equations, we generated skeleton equations for the second "general" experiment from the 12 Nguyen template equations that involve up to 3 variables and the 98 AI Feynman equations that include equations of up to 9 variables.

In each experimental setting, SymbolicGPT was trained using a minimum of 10,000 randomly generated symbolic regression instances belonging to the associated dimensional configuration, each consisting of an input dataset and an equation label. A further 1000 dataset-equation pairs were generated as the validation set, and 1000 new dataset-equation pairs were generated for the test set using different seeds.

We compared our methods with four existing models for nonlinear regression including Neural Symbolic Regression that Scales (**NeSymReS**)Biggio et al. (2021), Deep Symbolic Regression (**DSR**) Petersen et al. (2021), Genetic Programming (**GP**), and Multi-Layer Perceptrons (**MLP**).

For each method, we evaluated its performance on 1000 test instances of symbolic regression in each of the four experiment settings, using MSE_N as the fitness metric. We summarized the results in the cumulative distribution plots of Figure 2, showing the proportion of the test cases that attained error less than any given threshold value. Methods corresponding to curves positioned higher in the plot achieved higher accuracy on more test equations, and hence are better regressors. However, the most important region of the plot is the far left side, as the number of test cases that achieved the lowest possible error is an indication of how often the method would find a highly accurate fitting equation. As in the NeSymRes paper Biggio et al. (2021), we report NeSymRes for up to three variables.

5 Conclusions

In this work, we have presented a method that pushes the boundaries of language models and approaches the problem of symbolic regression from a new and powerful direction. We have employed language models in a novel way and with a novel approach, combining them with symbolic mathematics and order-invariant representations of point clouds. Our approach eliminates the per-instance computation expense of most regression methods and resolves the input restrictions imposed by other language-based regression models.

References

Douglas Adriano Augusto and Helio JC Barbosa. 2000. Symbolic regression via genetic programming. In *Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on*, pages 173–178. IEEE.

Luca Biggio, Tommaso Bendinelli, Aurelien Lucchi, and Giambattista Parascandolo. 2020. A seq2seq approach to symbolic regression. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. 2021. Neural symbolic regression that scales. *arXiv preprint arXiv:2106.06427*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Gang Chen. 2020. Learning symbolic expressions via gumbel-max equation learner network. *arXiv preprint arXiv:2012.06921*.

Roger Fletcher. 1987. *Practical Methods of Optimization*, second edition. John Wiley & Sons, New York, NY, USA.

Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. 2020. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*.

Michael Kommenda, Bogdan Burlacu, Gabriel Kronberger, and Michael Affenzeller. 2019. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, pages 1–31.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. *International Conference on Learning Representations*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.

Georg S Martius and Christoph Lampert. 2017. Extrapolation and learning equations. In *5th International Conference on Learning Representations, ICLR 2017-Workshop Track Proceedings*.

Ben McKay, Mark J Willis, and Geoffrey W Barton. 1995. Using a tree structured genetic algorithm to perform symbolic regression. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, pages 487–492. IET.

A Murari, E Peluso, M Gelfusa, I Lupelli, M Lungaroni, and P Gaudio. 2014. Symbolic regression via genetic programming for data driven derivation of confinement scaling laws without any assumption on their mathematical form. *Plasma Physics and Controlled Fusion*, 57(1):014008.

Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *preprint*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Subham Sahoo, Christoph Lampert, and Georg Martius. 2018. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR.

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

Silviu-Marian Udrescu and Max Tegmark. 2020. AI feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631.

Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. 2011. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119.

Yiqun Wang, Nicholas Wagner, and James M Rondinelli. 2019. Symbolic regression in materials science. *MRS Communications*, 9(3):793–805.