Symbolic Regression on Latent Representations from Autoencoders Using Genetic Programming

Nathan Anecone

October 11, 2024

Abstract

This paper presents a novel hybrid approach that combines autoencoder-based latent representations with symbolic regression using genetic programming to reconstruct the autoencoder's latent variables as independently executable symbolic equations from scratch. The evaluation is performed using the California Housing dataset. We compare the performance of the symbolic model with that of the autoencoder in terms of mean squared error (MSE) and inference time, showcasing the advantages of both methods. The method presented differs from traditional symbolic regression in that it generates equations. We close with a discussion of how these results show the promise of deriving the function approximated by neural network models and the potential value of utilizing evolutionary algorithms for synthetic data generation in machine learning contexts.

1 Introduction

Recent work has demonstrated the promise of hybrid architectures[1] in general and neuro-symbolic architectures specifically[2]. Progress in symbolic models such as State Space Models (SSMS) show the value of equation-driven generative models[3]. Research into meta-networks, which use the internal data generated inside an neural network as input for another system[4], presents an opportunity to derive the functions implicitly approximated by

the state of the input network according to the Universal Approximation Theorem (UAT) by use of a method that learns the representation inherent in the latent variables and generates a complementary, functionally approximate, mathematical formula. These derived mathematical formula can then serve as an effective proxy for the neural network's learned state and can be applied to the same domain when provided to a symbolic model on the same data.

The method demonstrated is unique in that it leverages evolutionary algorithms to generate equations by exploring mathematical problem space using a set of primitive operators until a solution is discovered that best matches the values held in the latent space. More traditional approaches to symbolic regression, such as

Autoencoders have shown great promise in dimensionality reduction and latent space representation. As such, they constitute an appealing candidate for providing a simplified basis from which to derive equations approximating a neural network's learned states. However, the latent variables produced by these models are often difficult to interpret and are not presented in a syntactically organized form. This paper proposes a method to reconstruct these latent representations using symbolic regression and genetic programming. The result transforms the model's learned features into a near functionally equivalent symbolic models. Our approach is tested on the California Housing dataset, and performance is compared in terms of MSE and inference time.

2 Methodology

2.1 Dataset

The dataset used for this experiment is scikit-learn's California Housing dataset. This dataset contains features such as house age and average number of rooms, and prices. This dataset was chosen for it being well-defined and maintained. Additionally the choice of features (house age, average number of rooms) was arbitrary for the purposes of the proof of concept described in this paper. Housing prices was selected as the target variable.

2.2 Autoencoder

An autoencoder model with 2 latent dimensions was trained on the dataset using features *HouseAge* and *AveRooms*. The autoencoder compresses the input features into a lower-dimensional latent space and then reconstructs the original data from this space. An autoencoder was specifically chosen as a piece of architecture for its capacity for dimensional reduction, as it is theorized that these reduced mappings are a more accessible candidate for symbolic regression than raw data.

2.3 Symbolic Regression

The latent representations from the autoencoder are used as the target variables for symbolic regression. Genetic programming, implemented using DEAP, is then applied to discover symbolic equations that approximate the latent variables.

DEAP is used for a genetic programming method for deriving the formulas primarily because, while unpredictable and often "crufty", evolutionary algorithms provide an open-ended and broadly combinatorial means for formula discovery for the widest possible variety of datasets.

We provide the genetic algorithm with a set of "safe" primitives for basic mathematical operators which prevent infinities and limit the search space to finite boundaries whilst providing maximum compositional possibilities for equation discovery.

3 Results

3.1 Mean Squared Error (MSE)

The MSE between the autoencoder's latent variables and the symbolic regression model's predictions was calculated for both latent dimensions. Since the results are probabilistic, the figures shown below are the average of 10 trials.

- Latent Dimension 1: MSE = 0.115218986668567
- Latent Dimension 2: MSE = 0.0422227188648843

3.2 Inference Time

We also compare the inference time between the two models:

- Symbolic Model Inference Time: 0.0064681 seconds
- Autoencoder Inference Time: 0.0004333 seconds

3.3 Symbolic Equation Example Output

Below is an example of a typical output for symbolic equations discovered by the procedure:

```
safe_log(subtract(safe_sqrt(safe_div(sin(add(safe_log(sin(sin(sin(sin(AveRooms)))))),
sin(safe_sqrt(multiply(sin(sin(sin(sin(sin(sin(safe_log(AveRooms)))))))), AveRooms))))),
safe_sqrt(sin(sin(multiply(safe_sqrt(AveRooms),
safe_div(AveRooms, HouseAge))))))), multiply(safe_exp(sin(
safe_log(sin(sin(sin(sin(sin(sin(sin(sin(sin(sin(AveRooms)))))))))))),
multiply(safe_exp(
sin(safe_log(sin(sin(sin(sin(sin(sin(sin(sin(sin(sin(AveRooms)))))))))))),
multiply(safe_exp(sin(safe_log(
sin(sin(sin(sin(sin(sin(sin(sin(sin(sin(AveRooms)))))))))),
subtract(safe_sqrt(sin(sin(sin(sin(sin(AveRooms)))))),
safe_exp(sin(sin(sin(sin(add(add(safe_log(sin(sin(sin(AveRooms)))),
sin(AveRooms)), safe_exp(safe_log(AveRooms)))))))))))))
```

It's worth noting that while this equation is inelegant and full of potentially redundant calculations, it had a MSE of 0.0033199708355414062 for the latent dimension it maps. This makes it an almost perfect functional reconstruction of the latent representation to which it corresponds. It should also be noted that while this rough approximation is not very human-readable, it is a drastic improvement over the raw latent representations in that regard. This also provides a unique window to interface traditional digital computing with the contents of a neural network.

4 Discussion

The results demonstrate that the symbolic model achieves comparable accuracy to the autoencoder in terms of MSE, particularly for latent dimension

2, where the error is quite low. However, the symbolic model's inference time is significantly slower due to the complexity of the equations generated by genetic programming.

The use of symbolic regression allows for interpretable models, which can provide insights into the relationships between input features and latent variables.

The present use of genetic programming to derive the equations can be likened to a rough first pass on a more refined future strategy that collects sample equations and further iterates on to select for parsimony and performance. As a proof of concept it succeeds in deriving functionally comparable equations for the latent representations. While the derived equations can tend toward the verbose (in line with genetic algorithm's general tendency to produce bloat), it nevertheless arrives at a decent approximation of the latent representation in up to 200 generations.

According to the UAT, a shallow neural network of appropriate size can approximate any continuous function[5]. From a symbolic modeling perspective, such a neural network can be viewed as a means to reconstitute the function from the learned parameters in the absence of prior knowledge about it. If we already had the function, it would be more practical in some cases to simply apply it to the target dataset in lieu of the more intensive network learning operations. Developments in "liquid time-constant networks" demonstrate the promise of generative symbolic models that apply structured operators (differential equations), to inputs that perform as universal approximators[6]. The present work offers a different approach to deriving a reconstructed approximation of the implicit function represented by the autoencoder's latent dimensions using genetic programming. While the current proposed method does not deliver the mathematically "ideal" function for the learned target space, further abstraction or retraining on the discovered equations could yield greater refinements which approach that goal.

5 Future Work

One of the implications of this project is the possibility of utilizing evolutionary algorithms for synthetic data generation in machine learning. The equations (or data generally construed) generated by the algorithm can be captured and retained and serve as the input to future iterative refinements. The stochastic "grow and prune" logic of evolutionary algorithms naturally

produces data that resists overfitting and underfitting, since overfitted and underfitted candidates can be pruned from the gene pool. It is biased only according to explicit selection metrics that the programmer can freely choose. Indeed, when combined with the outputs of a machine learning workflow the selection mechanisms of genetic programming invites a new way to approach fine tuning that combines reinforcement learning and the concept of adaptation to environmental conditions. In a programming context, "environmental conditions" means constraints specified by the developer.

While it remains speculative, it could be postulated that such a stochastic mechanism could yield a means out of "model collapse" by providing a mechanism to bootstrap a model on its own data by presenting it back to itself in a stochastic augmented form. Advancements using random noise in stable diffusion models hint at the potential of integrating stochastic processes into machine learning workflows[7]. To this end, we propose maintaining a running list of best individuals from evolutionary runs and using them as training data on future runs until increasingly improved candidates are evolved and trained on the system.

6 Conclusion

This paper presented a novel approach to using symbolic regression on latent representations from autoencoders to derive functionally approximate equations using genetic programming. Some future work may involve leveraging this methodology for the iterative improvement of the derived equations.

7 Code

The code for the autoencoder and symbolic regression implementation can be found at https://github.com/N8sGit/evolutionary_symbolic_regression_model

References

[1] M. Poli, A. W. Thomas, E. Nguyen, P. Ponnusamy, B. Deiseroth, K. Kersting, T. Suzuki, B. Hie, S. Ermon, C. Ré, C. Zhang, and S. Massaroli, "Mechanistic design and scaling of hybrid architectures," *arXiv preprint arXiv:2403.17844v2*, 2024. [Online]. Available: https://arxiv.org/abs/2403.17844v2

[2] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving olympiad geometry without human demonstrations," Nature, vol. 625, no. 7967, pp. 476-481, Jan. 2024. doi: 10.1038/s41586-023-06747-5.

[3] J. T. H. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," in *Proc. International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://arxiv.org/abs/2208.04933v3

[4] D. Lim, H. Maron, M. T. Law, J. Lorraine, and J. Lucas, "Graph Metanetworks for Processing Diverse Neural Architectures," *arXiv preprint arXiv:2312.04501v2*, Dec. 2023. [Online]. Available: https://arxiv.org/abs/2312.04501v2

[5] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. Neural networks, 2(3):183–192, 1989.

[6] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid Time-Constant Networks," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021. [Online]. Available: https://arxiv.org/abs/2004.08867v3 [6]M. Chen, Z. Zhou, J. Li, and Y. Wu, "An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and Optimization," arXiv, Apr. 2024. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2404.07771.

[7] M. Chen, Z. Zhou, J. Li, and Y. Wu, "An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and Optimization," arXiv, Apr. 2024. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2404.07771.