Active Learning Improves Performance on Symbolic Regression Tasks in StackGP

Nathan Haut
Michigan State University
East Lansing, Michigan, USA

Wolfgang Banzhaf
Michigan State University
East Lansing, Michigan, USA

Bill Punch
Michigan State University
East Lansing, Michigan, USA

ABSTRACT

This paper introduces an active learning method for symbolic regression using StackGP. The approach begins with a small number of data points for StackGP to model. To improve the model the system incrementally adds the data point characterized by maximizing prediction uncertainty as measured by the model ensemble. Symbolic regression is re-run with the larger data set. This cycle continues until the system satisfies a termination criterion. The Feynman AI benchmark set of equations is used to examine the ability of the method to find appropriate models using as few data points as possible. The approach successfully rediscovered 72 of the 100 Feynman equations without the use of domain expertise or data translation.

CCS CONCEPTS

• **Computing methodologies** → **Representation of mathematical functions**; **Supervised learning by regression**; **Genetic programming**; **Active learning settings**.

KEYWORDS

active learning, symbolic regression, genetic programming

ACM Reference Format:
Nathan Haut, Wolfgang Banzhaf, and Bill Punch. 2022. Active Learning Improves Performance on Symbolic Regression Tasks in StackGP. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528941

1 INTRODUCTION

Active learning [1] is a machine learning strategy where an algorithm self-selects additional training data to maximally inform its learning process. Active learning has been applied to genetic programming classification tasks where data are only labelled when the developing models encounter data that can't be classified [3]. This was found to reduce the total effort needed to label training data, since only a subset had to be labelled before finding accurate models. Active learning has also been applied to genetic programming where training sets are large by selecting sub-samples of the training data to be used. Active learning for sub-sampling was found to decrease training times to find quality binary classification models by an order of magnitude [2].

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528941

The goal of StackGP with active learning is to create a general purpose GP system that requires no domain expertise, uses as few data points as possible, and guides data collection to be maximally informative. Beyond data collection for model training, the developed models could be used to design experiments to further explore the system of study. As well, the models could be used to accelerate the development process by recommending target conditions for the system of study.

2 METHODS

Our active learning strategy is an iterative process that trains models on data, selects an ensemble of good models, then uses the ensemble to find a new point of maximum uncertainty to add to the training data . The algorithm is summarized in Algorithm 1 and each part is described in detail in the subsection following.

StackGP is the stack based genetic programming system used to evolve models during the model development step of the active learning strategy [4]. We implemented StackGP within Mathematica for this work. The parameters used are shown in Table 1.

2.1 Active Learning

The goal of active learning is to strategically select new data points that are most informative to current models. One way to identify informative points is to find disagreement among current models. The uncertainty metric Δ we apply to quantify disagreement is defined as the standard deviation of the ensemble divided by the 70 percent trimmed mean of the absolute value of ensemble responses.

$$\Delta = \frac{\text{Std}\left(\text{ EnsembleResponses }\right)}{\text{TrimmedMean}(\text{Abs}((\text{ EnsembleResponses}, 0.3))}$$

The trimmed mean is used to ignore potentially asymptotic behavior that could occur in a few of the models. Below is the step-by-step explanation of how this active learning approach works.

2.1.1 Initialization. To start, 3 random data points from the region defined in the benchmark set [7, 8] are generated. Another 100 data points are generated as test points and are used purely for tracking the progress of model development. They are not used to inform model development.

An initial set of models are trained on these 3 data points. Evolution is allowed to run for up to 2 minutes running independently on 4 cores. Each run has a population size of 300 models initialized randomly at the outset, such that the starting population consists of random models with an operator stack of 10 operators or less. An operator is any of the math operators allowed to be used during evolution plus the pop operator. The math operators used are: $+, -, *, /, Exp, Sqrt, Inverse, Squared, Sin, Cos, Tan, ArcSin, ArcCos, ArcTan, TanH$ and Ln.

Algorithm 1 Active Learning Process

```
TrainingData ← 3StartingPoints                          ▷ Generate initial random training data
Models ← RandomModels                                   ▷ Generate initial random models
Models ← Evolve(TrainingData, Models)                   ▷ Train models on starting data
while BestModelError ≠ 0 do                             ▷ While perfect model not found
    Ensemble ← EnsembleSelect(Models).                  ▷ Select ensemble of models
    NewPoint ← MaximizeUncertainty(Ensemble)            ▷ Find point that maximizes uncertainty
    if NewPoint ⊂ TrainingPoints then                   ▷ If point already selected
        NewPoint ← MaximizeUncertainty(SubSpace(Ensemble))   ▷ Search a subspace
    end if
    TrainingData ← Append(TrainingData, NewPoint)       ▷ Add new point to training data
    Models ← Evolve(TrainingData, Models)               ▷ Evolve new models with new data using best models to seed evolution
end while
```

2.1.2 Evolution Epochs. The evolution process relies on a multi-objective fitness function that utilizes Pareto optimality of correlation and complexity. The Pareto front represents the models with the best trade-off between correlation and complexity.

The models are evolved using crossover and mutation. Crossover employs a two point crossover operator modified for the stack data structure. Mutation allows for 6 different types of mutation operators with uniform probability: mutating variables or constants, mutating math operators, pushing new variables and operators to the top of the stacks, trimming off the bottom of the stack, pushing new variables and constants to the bottom of the stacks, and inserting new operators at a random position in the stack.

2.1.3 Ensemble Generation and Data Selection. Once the models are developed, an ensemble is generated using the data balancing ensemble approach described in [5]. Early in the active learning process ensembles will be smaller since the number of data clusters is limited by the number of data points. As the number of data points increases, the limit on the number of clusters increases. The maximum number of clusters is capped at 10 to ensure ensembles do not grow beyond 10 models. This helps prevent ensemble evaluation from becoming too computationally costly. In the event that all data points are similar and only a single cluster forms, the Pareto front of the models is chosen as the ensemble rather than a single model.

Using the selected ensemble, Mathematica's NMaximize[1] [6] function is employed to find the data point that maximizes the uncertainty metric defined in section 3.1, within the bounds of each variable as described in the Feynman Symbolic Regression Database [7]. It is possible that a local maximum is found rather than a global maximum. This is acceptable since such a point still has a relatively high uncertainty. The parameters found to maximize uncertainty are then used to collect the true model response. This data is then added to the training set and will be used in the next run of model evolution.

It is possible a point that already exists in the training set is selected as the new point. Rather than duplicating a point, however, a new point is selected by maximizing the uncertainty of the ensemble in a random region of the original search space. This helps ensure that new information is being added in each iteration.

Once the new training point has been added to the training set, another evolutionary epoch begins. The new evolutionary epoch is seeded with 20% of models nearest to the Pareto front of the previous epoch. This ensures that good models are not lost between evolutionary epochs. It does, however, introduce the risk that these more developed models will dominate less developed (or random) models at the beginning of the new epoch and thus biases evolution. We limit this risk by choosing small tournament sizes, although it could be further limited in the future using other methods such as age layering.

The learning process is repeated until a perfect model is found or a maximum number of iterations has completed.

Table 1: StackGP & Active learning Parameter Settings

Parameter	Setting
Mutation Rate	79
Crossover Rate	11
Spawn Rate	10
Elitism Rate	10
Crossover Method	2 Pt.
Tournament Size	5
Population Size	300
Selection Rate	20
Selection Method	Pareto Tournaments
Fitness Measures	Correlation vs. Simplicity

3 RESULTS

37 of the 100 equations we were able to solve with just the initial random 3 data points. The minimum number of points needed by AIFeynman was 10, so StackGP outperformed AIFeynman on all of these problems. It also indicates that these problems are trivially solvable and active learning is not necessary. Of the remaining problems, 16 were solved using fewer data points than what was reported by AIFeynman. For these problems, it seems that the active learning approach had a positive effect on search success. One of the equations needed the same number of points as AIFeynman. 18

[1]Mathematica's NMaximize function was used with default settings which allows Mathematica to choose a maximization method from the following options: Nelder Mead, Differential Evolution, Simulated Annealing, and Random Search.

Table 2: StackGP with Active Learning Performance Summary

Performance	Total Equations
Trivial (Only 3 Points Needed)	37
Outperformed AIFeynman	16
Underperformed AIFeynman	18
Matched AIFeynman	1
Failed to Solve	28

of the problems required more data points than was reported by AIFeynman. The 28 remaining problems were not solved within 100 iterations of active learning and so it is not possible to compare the effect that active learning had on the success of those searches. The results are summarized in Table 2.

According to Udrescu and Tegmark, Eureqa is the best available commercial symbolic regression software [8]. Eureqa was found to solve 71 of the 100 Feynman equations using 300 data points and 2 hours of compute time for each equation. StackGP with active learning was able to find 72 of the 100 Feynman equations demonstrating a similar performance as Eureqa, although not all the same equations were solved.

The performance on each individual equation is reported in [4]. The formulae for each equation number alongside the variable ranges and sample data can be found in the Feynman Symbolic Regression Database [7] where they are ordered in the same way. In the following we discuss a few examples.

Equation number 22 is an example of a problem that needed just 3 points to be found:

$$\tau = rF\sin(\theta) \tag{Eq 22}$$

Looking at the equation we can see that it is relatively simple and would require only 3 operators (sin, *, *) and 3 variables (r, f, θ). It is likely easy to find, both due to its simplicity and since the terms are combined as products, which makes each variable's contribution to the response data easily distinguishable and similar in magnitude.

Equation number 3 is an example of an equation where the active learning approach with StackGP outperformed AIFeynman, needing just 42.5 points on average compared to the 1000 points needed by AIFeynman:

$$f = \frac{e^{-\frac{1}{2}\left(\frac{\theta-\theta_1}{\sigma}\right)^2}}{\sqrt{(2\pi)}\sigma} \tag{Eq 3}$$

It bears mentioning that this equation was unsolvable by Eureqa.

Equation number 5 is an example of an equation that was unsolvable by StackGP with active learning and by Eureqa:

$$F = \frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2} \tag{Eq 5}$$

It required 1 million data points to be solved by AIFeynman. This equation is rather complicated since it has 9 variables and the contributions of each variable to the response are vastly different depending on where they are in the equation.

4 ABLATION STUDY

An ablation study was completed to determine if the performance of StackGP using active learning can be attributed to the active learning strategy or the stack based genetic programming system. The ablation study compared the active learning approach against a modified approach where new data points were randomly selected rather than selected according to the active learning strategy. A sample of equations from the set were chosen to examine a variety of equation forms. Specifically, equations 1, 2, and 3 were chosen to highlight how very similar but slightly modified equations can differ in how successful this active learning approach is.

For comparison purposes, equation 1, 2, and 3 are shown below. A variable is added between each equation, making equation 2 slightly more complex than equation 1, and equation 3 slightly more complex than equation 2:

$$f = \frac{e^{-\frac{\theta^2}{2}}}{\sqrt{(2\pi)}} \tag{Eq 1}$$

$$f = \frac{e^{-\frac{1}{2}\left(\frac{\theta}{\sigma}\right)^2}}{\sqrt{2\pi}\sigma} \tag{Eq 2}$$

$$f = \frac{e^{-\frac{1}{2}\left(\frac{\theta-\theta_1}{\sigma}\right)^2}}{\sqrt{(2\pi)}\sigma} \tag{Eq 3}$$

The expected behaviour would be for the equations to be more difficult to find as the complexity increases. When using random point selection, this behaviour is observed, but when using active learning, equation 2 seems to be more difficult to find than equation 3. It is unexpected that active learning works as well as it does on equation 3, when it performs worse than random point selection on equation 2, despite equation 2 being similar yet simpler than equation 3. It is also unclear what would make equation 2 more difficult for active learning than random point selection, so further analysis is planned for the future.

Equation 24 is another example of an equation that active learning performed well on and excelled over random point selection:

$$E = \frac{1}{4}mx^2\left(\omega^2+\omega 1^2\right) \tag{Eq 24}$$

Equation 14 and Equation 47 both showed worse performance using active learning over random search:

$$U = Gm_1m_2\left(\frac{1}{r_2}-\frac{1}{r_1}\right) \tag{Eq 14}$$

$$\kappa = \frac{kv}{A(\gamma-1)} \tag{Eq 47}$$

It is possible that the active learning point selection is misled by these equations to select points that are not maximally informative or points that are very similar to points previously selected.

Table 3 summarizes the results of the ablation study on a selected number of equations from the AIFeynman benchmark set.

Table 3: Average number of points needed in Active Learning (AL) vs. Random Point Selection (Random). Number of trials.

EQ#	AL	Random	Trials
1	3	3	100
2	43	28.5	100
3	42.5	>202	100
4	25	26	100
10	4	5	100
11	3	3	100
12	3	3	100
14	19.5	14	100
15	3	3	100
16	3	3	100
23	4	4	100
24	28.5	49.5	100
32	10.5	11	100
47	28.5	17.5	40
60	8	7.5	100
61	30	30.5	40

5 DISCUSSION

Although the active learning approach did not outperform AIFeynman on all equations tested, it does show promise in that it has a similar success rate to Eureqa and uses no domain expertise, unlike AIFeynman. The active learning approach represents a self-guided experimentation process where the machine learning algorithm can direct an experimentation and design process so that researchers can spend less time planning their next experiments. Even more importantly, with active learning, it is less likely that an exhaustive set of experiments needs to be completed to fully understand a system of study.

It was observed in the ablation study that some types of problems are better suited for this method of active learning while other types of problems are more difficult than random point selection. For those problems that are more difficult it is possible that the active learning point selection was misled to choose points that are not actually maximally informative. An attempt to avoid this problem was made but was not sufficient to ensure similar points are not repeatedly selected. Further work will explore an approach where new points have to be a minimum distance away from points already in the data set to ensure that similar points are not gathered at every selection event. Alternatively, a hybrid random and informed point selection could be utilized.

A second possibility is that the issue lies with the uncertainty metric used. The uncertainty metric used is a *relative* measure since it is scaled by the magnitude of the ensemble response. This seemed like a good approach since the magnitude of uncertainty would likely increase as the magnitude of the response increases, but that may not always be the case. If it is not the case, point selection could become biased towards regions where the uncertainty metric is magnified by the smaller ensemble response. This could potentially be fixed by changing the uncertainty metric to not be relative to the magnitude of the ensemble response.

Further research to explore how well various uncertainty metrics affect the success of this active learning approach on various problem types will be useful. As well, it could be beneficial to explore and classify the types of problems that tend to be difficult or easy for active learning.

From observation it seems that of all the equations in the Feynman data set, the ones that tend to pose difficulty for this active learning approach tend to have complex denominators. This could support the concern that the relative uncertainty metric is being misled for some problem types. Alternatively, it could highlight a weakness with current symbolic regression implementations since Eureqa struggled with many of those problems as well. It is possible that when variables exist in the denominators of problems, that it becomes more difficult for symbolic regression to determine the true contribution of those variables. To determine if this is a larger scale weakness with symbolic regression it could be useful to compare several additional symbolic regression implementations on those difficult problems.

Future research is planned to explore the applications of other active learning techniques, both model and data driven, to genetic programming with the goal of determining which methods are most successful for different types of problems. This information can form the basis for an active learning toolkit that can be used by researchers in various fields to accelerate their data collection process.

ACKNOWLEDGMENTS

Funding by the John R. Koza Endowment to Michigan State University is gratefully acknowledged. MSU's iCER High-Performance Computing Center provided the compute cycles for this research.

REFERENCES

[1] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research* 4, 1 (1996), 129–145.

[2] Robert Curry, Peter Lichodzijewski, and Malcolm I Heywood. 2007. Scaling Genetic Programming to Large Datasets Using Hierarchical Dynamic Subset Selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37, 4 (2007), 1065–1073.

[3] Junio De Freitas, Gisele L Pappa, Altigran S da Silva, Marcos A Goncales, Edleno Moura, Adriano Veloso, Alberto HF Laender, and Moisés G de Carvalho. 2010. Active Learning Genetic Programming for Record Deduplication. In *IEEE Congress on Evolutionary Computation*. 1–8.

[4] Nathan Haut, Wolfgang Banzhaf, and Bill Punch. 2022. Active Learning Improves Performance on Symbolic Regression Tasks in StackGP. *arXiv* 2202.04708 (2022).

[5] Mark Kotanchek and Nathan Haut. 2022. Back To The Future — Revisiting OrdinalGP & Trustable Models After a Decade. In *Genetic Programming Theory and Practice XVIII*, Wolfgang Banzhaf, Leonardo Trujillo, Stephan Winkler, and Bill Worzel (Eds.). Springer, Singapore, 129–142.

[6] Wolfram Research. 13. NMaximize. https://reference.wolfram.com/language/ref/NMaximize.html. [version 13.0].

[7] Max Tegmark. [n.d.]. *Welcome to the Feynman Symbolic Regression Database!* Retrieved January 26, 2022 from https://space.mit.edu/home/tegmark/aifeynman.html#:~:text=As%20opposed%20to%20linear%20regression,any%20combination%20of%20mathematical%20symbols.

[8] Silviu-Marian Udrescu and Max Tegmark. 2020. A physics-inspired method for symbolic regression. *Science Advances* 6 (2020), eaay2631.