Active Learning Informs Symbolic Regression Model Development in Genetic Programming

Nathan Haut
hautnath@msu.edu
Michigan State University
East Lansing, Michigan, USA

Bill Punch
Michigan State University
East Lansing, Michigan, USA
punch@msu.edu

Wolfgang Banzhaf
Michigan State University
East Lansing, Michigan, USA
banzhafw@msu.edu

ABSTRACT

Active learning for genetic programming using model ensemble uncertainty was explored across a range of uncertainty metrics to determine if active learning can be used with GP to minimize training set sizes by selecting maximally informative samples to guide evolution. The choice of uncertainty metric was found to have a significant impact on the success of active learning to inform model development in genetic programming. Differential evolution was found to be an effective optimizer, likely due to the non-convex nature of the uncertainty space, while differential entropy was found to be an effective uncertainty metric. Uncertainty-based active learning was compared to two random sampling methods and the results show that active learning successfully identified informative samples and can be used with GP to reduce required training set sizes to arrive at a solution.

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, Bill Punch, and Wolfgang Banzhaf. 2023. Active Learning Informs Symbolic Regression Model Development in Genetic Programming. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590577

1 INTRODUCTION

Active learning is a method used in conjunction with machine learning to actively select new training data with the goal of selecting data points that will maximally inform the machine learning model [1]. Various forms of active learning (AL) exist, with three types dominating: pool-based AL, stream-based AL, and membership query synthesis [5]. Pool-based and stream-based methods both have a set of training samples to choose from, with the goal of selecting and training on only a small subset of maximally informative cases. The key difference between stream and pool-based methods is that stream-based methods check each potential training case in order one by one and only admit them to the training set if data points are "informative". Pool-based methods differ by searching a set of data points for the ones that are most informative. Membership query synthesis approaches do not have a set of already existing training samples to choose from, instead, they search a training space to find and synthesize new training data points that are expected to maximally inform the machine learning model. Once synthesized, a new data point is then labelled by the researcher via experimentation or expert knowledge.

In this contribution, we apply active learning strategies for genetic programming used in symbolic regression tasks. The goal is to exploit some of the features of GP, in particular its reliance on a population of models. More specifically, we want to utilize uncertainty and diversity measures to accelerate the discovery of models (physics equations in our study). The idea is to look for disagreement among high-quality individuals in the population as a guide to locate informative data points to add to the training set and thus reduce the required size of the training set.

2 METHODS

2.1 Active Learning

We use uncertainty-based active learning for symbolic regression with GP to accelerate the development of models for the Feynman Symbolic Regression Dataset [6]. Uncertainty-based AL utilizes an ensemble of diverse, high-quality models from a population to search for regions in the search space with high uncertainty or disagreement between the models.

Several different uncertainty metrics are implemented to determine their respective impact on the success of the task. Success of active learning by maximizing uncertainty would indicate that the diversity of the population can be exploited to guide the collection of informative data.

The GP implementation used was the python version of StackGP [3] and is available here [2] along with the active learning code. The parameters used to run the GP system with active learning are shown in Table 1. Note that the tournament selection method used is Pareto tournament selection [4], where accuracy (R^2) and complexity (stack lengths) are the two objectives.

2.1.1 Maximizing Uncertainty. Several different uncertainty metrics were explored to determine how different measures impact the success of active learning. Success is determined by how few training points are required to find a solution for each problem. As an overview, each approach begins by selecting an ensemble of models,

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590577

then a function that uses the specific uncertainty metric along with the ensemble and current training set is created. This function is then fed to an optimizer to search for regions of relatively high uncertainty. The most uncertain point found is then returned and added to the training set. In total there were 6 different approaches tested which varied in how they quantified disagreement, whether outlier predictions are considered, and which optimizer was used. The steps and methods will be described in greater detail below and the entire process is depicted in Algorithm 2.

Generating the ensemble is the first step in uncertainty-based active learning. The goals for generating the ensemble were to capture diverse, high-quality individuals from the population while keeping the size of the ensemble relatively small so that the computational cost of optimizing uncertainty is reasonable. The diversity goal is essential to the success of active learning since disagreement between models is a necessary requirement for selecting informative points to minimize training set size. The method chosen to capture both diversity and quality from the model population works by clustering the training data using the input space and selecting a model that best fits each cluster, ensuring no model is selected more than once. If a model is already selected by another cluster, the next best unselected model is chosen. The minimum number of clusters is set to 3 and the maximum is set to 10. Thus, 3-10 models are chosen for inclusion in an ensemble. Data clustering was chosen with the intent to capture diversity by focusing on models that have biases for different regions of the training space. Quality in the population would be captured since only models with the best fitness were selected for each cluster. The algorithm to generate the ensemble is described in detail in Algorithm 1.

Algorithm 1 Ensemble generation process to select diverse high-quality models.

```
procedure EnsembleSelect(models,trainingData,responseData)
    selectedModels ← [ ]                                  ▷ Initialize ensemble
    nClusters ← min(len(trainingData), 10)                ▷ Determine number of clusters
    clusters ← KMeans(nClusters).fit_predict(trainingData)
    for i = 0;i++; i < nClusters do                       ▷ Loop over data clusters
        modelErrors ← computeError(models, clusters[i])
        sortedModels ← sortBy(models, modelErrors)
        j = 0
        while sortedModels[j] in selectedModels do        ▷ Find best unselected model
            j++
        end while
        selectedModels = join(selectedModels, sortedModels[j]   ▷ Add to ensemble
    end for
    return selectedModels                                 ▷ Return ensemble
end procedure
```

The second step of this method is to utilize the specified uncertainty function with both the current training data and the selected ensemble to find a new point with high information content to add to the training set. The optimizer is given the uncertainty function and search space boundaries to find a point of relatively high uncertainty. In the case that an already selected point is re-selected, a new search is initiated within a random sub-region until a unique point is selected. This ensures that new information is added in each iteration.

In total 5 different uncertainty metrics were used, shown by Equations 1 to 5. The two methods used for optimization were Scipy Optimize's minimize and differential evolution functions.

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
Parallel Runs	4
Generations	1000

$$\Delta = \frac{\text{Std(EnsembleResponses)}}{\text{Mean(Abs(EnsembleResponses))}} \tag{1}$$

$$\Delta = \frac{\text{TrimmedStd(EnsembleResponses, 0.3)}}{\text{TrimmedMean(Abs(EnsembleResponses), 0.3)}} \tag{2}$$

$$\Delta = \frac{\text{Std(EnsembleResponses)}}{\text{TrimmedMean(Abs(EnsembleResponses), 0.3)}} \tag{3}$$

$$\Delta = \text{Std(EnsembleResponses)} \tag{4}$$

$$\Delta = \text{DifferentialEntropy(EnsembleResponses)} \tag{5}$$

Algorithm 2 Active Learning Process Using Uncertainty

```
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 ← MaxUncertainty(Ensemble)           ▷ Find point of max uncertainty
    if NewPoint ⊂ TrainingData then               ▷ If point already selected
        NewPoint ← MaxUncertainty(SubSpace(Ensemble))   ▷ Search a subspace
    end if
    TrainingData ← Append(TrainingData, NewPoint)     ▷ Add new point
    Models ← Evolve(TrainingData, Models)         ▷ Evolve new models with new data using
best models to seed evolution
end while
```

2.1.2 Benchmark Testing. Each active learning approach was compared on a benchmark set of 35 out of the 100 equations from the Feynman Symbolic Regression Dataset [7]. Of the 100 problems, 37 are uninteresting for active learning since in a previous study it was found that those equations needed just 3 data points to be solved [3]. Of the remaining 63 problems, we chose 35 to test the efficacy of active learning, while keeping computational costs reasonable. These 35 problems were selected since initial testing without active learning showed they could frequently be solved with 4 to 1000 training points.

2.2 Random Sampling

As a baseline, we used random sampling of data points from uniform and normal distributions to determine if an active learning method improves learning progress over a naive sampling of training data.

To create a fair comparison against the active learning methods, a simple substitution was made where instead of using active learning

to maximize uncertainty a random point was added in each iteration. Beyond that substitution, the algorithm remains the same.

3 RESULTS & DISCUSSION

The results of comparing the different uncertainty-based active learning methods are shown in Table 2. The results in this table represent approximately 65,000 compute hours on Xeon 6148, 40 core CPUs (or their equivalent across the cluster). The table compares each method to both uniform and normally distributed random sampling. Each experiment was repeated 100 times and the median performance used for comparison. The data shows that the active learning approaches that rely on differential entropy perform best compared to the other uncertainty metrics, with differential entropy using differential evolution as the optimizer working best. This indicates that differential entropy as an uncertainty metric best correlates to the informativeness of selected data points. The fact that differential evolution improves the performance of active learning when using differential entropy indicates that the search space is likely very rigid or non-convex and difficult to search using other optimization strategies. It is also interesting to note that the simplest uncertainty metric, standard deviation, performs reasonably well, which makes it an appealing method since it is simple to implement and computationally light.

The success of active learning seems dependent on the problem. For example, active learning worked very well on the problem with the formulation

$$x_1 y_1 + x_2 y_2 + x_3 y_3 \tag{6}$$

where the uniform and normal random sampling approaches required 88.5 and 82 data points while all the active learning methods required significantly fewer ranging from 21 to 52.5 points. For many of the problems, where active learning did not outperform random sampling, the difference was not very significant, generally requiring only a few more points on average. There is one problem though with the formulation

$$n * Exp[(m * g * x)/(k * T)] \tag{7}$$

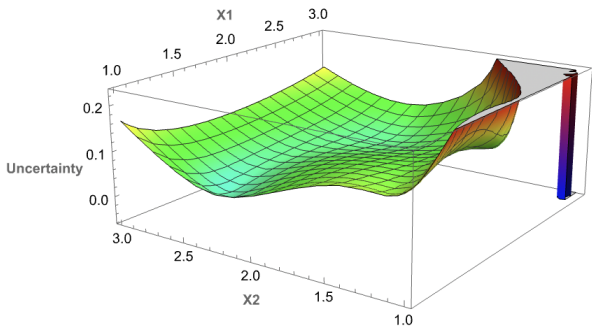
where normal random sampling actually performed best by a significant margin requiring just 82 points on average compared to the active learning methods which ranged from 144-876 pts on average. It is interesting to note that uniform sampling required 453 points, so all but one of the active learning methods did still outperform the uniform sampling on that problem. Looking at the formula it seems likely that the difficulty comes from the fact that the equation can produce responses of significantly different magnitudes considering there are 5 terms in the exponent.

Looking at the formulations for some of the uncertainty metrics we can see that it is possible to have asymptotic behavior in the uncertainty spaces when the mean ensemble prediction averages to 0 in Equations 1, 2, and 3. This could potentially lead to regions being marked as having high uncertainty in cases where all models in the ensemble actually correctly predict a value around 0. An example of this is shown in Figure 1, where asymptotic behavior is shown due to the mean prediction of the ensemble being 0. This could also potentially be why some of the active learning methods performed poorly on problem 14, which can evaluate to 0 anytime r_1 and r_2 are equal. The formulation for problem 14 is

$$G * m_1 * m_2 * (\frac{1}{r_1} - \frac{1}{r_2}). \tag{8}$$

These 3 metrics could possibly be improved by preventing the denominator from ever evaluating to 0 or values near 0. Maybe a threshold of 1 as a minimum could be included to avoid asymptotic behavior caused by the denominator while still allowing the uncertainty metric to be a measure of relative uncertainty. Equation 4 for example avoids the issue of 0 denominators but loses the potential benefit of being a *relative* uncertainty measure. We can see an example of an uncertainty space for the same problem (problem 2) when using Equation 4 instead of 3 in Figure 2. Asymptotic behavior is avoided here, but we can see another type of bias, which is the bias for high uncertainty near the borders of the training space as a result of models being unconstrained outside of the boundaries. This bias seems potentially unavoidable since uncertainty should always be high outside of the training space since the models have no knowledge of regions outside of the training space. A potential way to remedy this bias could be by implementing a secondary metric such as data diversity to be used for active learning, that would help ensure that regions of the training space aren't oversampled as a results of the biases of the uncertainty metrics.

Figure 1: A sample ensemble uncertainty space is shown using an ensemble selected from a model population trained for problem 2. Problem 2 has 2 input variables, represented in the figure as $x1$ and $x2$. The vertical axis represents the uncertainty. The red regions represent high uncertainty while blue regions represent low uncertainty. The uncertainty metric used to generate this was Equation 3. This figure demonstrates the possibility of asymptotic behavior (seen in the right corner), which could be misleading for the active learning process and lead to larger training sets. Note: Active learning is searching for the high uncertainties (red regions).



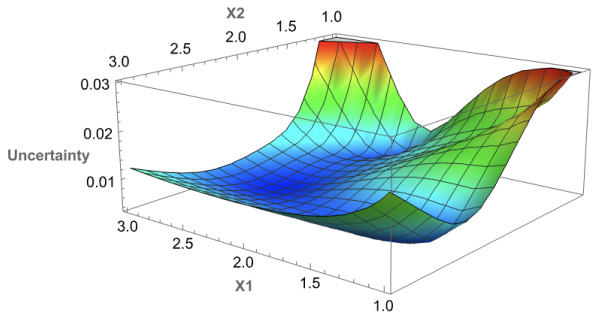
4 CONCLUSION

A total of 6 different uncertainty-based active learning methods were compared to determine how well each approach is able to maximize the information gain of selected training samples and reduce training set sizes. The results showed that active learning was consistently able to reduce the required size of training sets, with

Table 2: Shown are the performances of each approach compared to both uniform random sampling and normally distributed random sampling. The number indicated is the number of problems of 35 where the active learning method matched or performed better than the random sampling method being compared with. Each method is indicated by the equation number and an abbreviated form of the equation. Note (DE) indicates differential evolution was used as the optimizer instead of Scipy Minimize for that method.

	EQ 1 Std/Mean	EQ 2 TrStd/TrMean	EQ 3 Std/TrMean	EQ 4 Std	EQ 5 DE	EQ 5 (DE) DE (DE)
Compared to U. Sampling	19	20	24	27	31	33
Compared to N. Sampling	22	20	20	22	29	30

Figure 2: A sample ensemble uncertainty space is shown for the same problem as Figure 1. The red regions represent high uncertainty while blue regions represent low uncertainty. Some common trends can be seen here, such as higher uncertainty near the boundaries and the fact that the search space is non-convex. The uncertainty metric used was Equation 4.



differential entropy as an uncertainty metric outperforming the methods that rely on the standard deviation and mean predictions of the ensembles. This indicates that the measure of differential entropy best correlates to information gain when selecting new training samples. When using differential evolution as the optimizer with differential entropy we saw a slight improvement over using Scipy Optimize's minimize function indicating that differential evolution is better suited for exploring the ensemble uncertainty space. The bias for selecting points near the borders of the training regions was identified, which will lead to future work on how to avoid that bias to improve sampling. As well, the potential for the relative uncertainty metrics to display asymptotic behavior was identified as a result of ensemble mean predictions of 0 being included in the denominator. This is another potential bias that future work will address.

Further, the method for ensemble design could be reconsidered to potentially improve how ensemble uncertainty translates into informing development of model populations during the active learning iterations. The model ensemble method being used was developed with the goal of selecting diverse high-quality models from the population, with the intended use-case being to generate deployable models that will remain stable under new conditions. It could potentially improve the active learning process if an ensemble generation method was developed with active learning in mind and how the uncertainty of the ensemble would translate into informativeness for model development.

Overall, the synergy between active learning and genetic programming seems promising with these results showing that a good uncertainty metric can improve informativeness of training data while minimizing training set sizes. This is useful in scenarios where collecting data is expensive, so minimizing the need to collect data could accelerate the work while reducing costs. It could also be useful in scenarios where a large training sample is available but maybe heavily biased, so extracting only the interesting points would lead to developing models resistant to the bias of the data collection. Future work to improve the uncertainty metric and ensemble design could improve effectiveness of active learning. Genetic programming also seems to be a very natural fit for active learning since evolution can very easily be adapted to be interactive, which allows active learning to be integrated so evolution can both guide model development and collection of training samples simultaneously. This potential could lead to development of new applications where genetic programming systems could be used by researchers with dual purpose to both guide experimental design and develop models to provide insight into the system being studied.

ACKNOWLEDGMENTS

This work was supported in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled 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] Nathan Haut. [n. d.]. *StackGP.* Retrieved February 2, 2023 from https://github.com/hoolagans/StackGP
[3] Nathan Haut, Wolfgang Banzhaf, and Bill Punch. 2022. Active Learning Improves Performance on Symbolic Regression Tasks in StackGP. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 550–553.
[4] Mark Kotanchek, Guido Smits, and Ekaterina Vladislavleva. 2007. Pursuing The Pareto Paradigm: Tournaments, Algorithm Variations, And Ordinal Optimization. In *Genetic Programming Theory and Practice IV*, Rick Riolo, Terence Soule, and Bill Worzel (Eds.). Springer, 167–185.
[5] Burr Settles. 2009. *Active Learning Literature Survey.* Computer Sciences Technical Report 1648. University of Wisconsin–Madison.
[6] 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.
[7] S.-M. Udrescu and Tegmark M. 2020. A physics-inspired method for symbolic regression. *Science Advances* 6 (2020), eaay2631.