Research Internship Report
Bayesian Symbolic Regression

David Coba
St. no. 12439665
Research Master's Psychology
University of Amsterdam
Psychological Methods

Supervised by:
Don van den Bergh
Eric-Jan Wagenmakers

XX July 2021

Abstract

[To be edited] Symbolic regression is a machine learning method that generates explicit mathematical expressions by composing basic functions. Since the models are just mathematical expressions they are fully interpretable, unlike most other machine learning techniques. The goal of this project is to develop and test a general Bayesian symbolic regression framework. The current state of the art in symbolic regression are methods that are able to include information about the structure of the target system they are trying to model. However, they use an approach with neural networks that is convoluted and hard to generalize. We believe that Bayesian methods could be a straightforward alternative to incorporate prior knowledge.

Keywords: kw1, kw2, . . .

Research Internship Report
Bayesian Symbolic Regression

Introduction

Symbolic regression is a supervised machine learning technique that attempts to find mathematical expressions that describe relations between variables of a dataset. The mathematical expressions can be arbitrarily complex and are constructed from a prespecified set of features and operators (e.g. addition, multiplication, trigonometric functions, etc.). The main advantage of symbolic regression over other machine learning techniques is that the resulting models are explicit and interpretable. The goal of this internship was to implement an easy to use Bayesian symbolic regression program and to compare its performance against other methods.

Typical black-box machine learning models are hard to use in basic scientific research because they lack interpretability, but symbolic regression avoids this issue since the output is just a mathematical expression. This allows symbolic regression to be composable with any scientific model that is specified mathematically. For example, Dandekar et al. (2020) use symbolic regression and neural networks to extend an epidemiological model of COVID-19 spread (Lin et al., 2020). The model represents the spread of the pandemic as a system of differential equations, and captures our scientific knowledge about how contagious diseases spread. However, they did not have a good mathematical expression for a complex function that quantified the role of quarantine policies on the spread of the virus. With their approach they were able to recover—or discover—a mathematical expression for that function with a plausible scientific interpretation. This example illustrates the possibilities that symbolic regression techniques offer to bridge black-box modeling techniques with scientific modeling of complex systems. In recent years there has been a movement of researchers arguing in favor of doing more formal modeling in psychological research (e.g. Guest and Martin, 2020; Robinaugh et al., 2020; van Rooij and Baggio, 2020). And, specially in our discipline where we lack established formal models, symbolic regression seems like a relevant tool for the development of formal models of psychological systems.

The space of possible mathematical expressions for any given problem is infinitely large, and therefore it is not viable to explore it exhaustively. There are multiple methods that aim to recover mathematical expressions from a dataset, although they are not always called *symbolic regression* in the literature. Some of these methods use sparse regression (e.g. Brunton et al., 2016), neural networks (e.g. Sahoo et al., 2018) or a combination of both (e.g. Both et al., 2021). However, the most common approach to perform symbolic regression is to do a targeted search using evolutionary algorithms, which work by mimicking the evolution of a population of candidate expressions. Examples of this approach are the DEAP library (Fortin et al., 2012), the widely used proprietary software *Eureqa*[1] (Schmidt & Lipson, 2009) and the open-source implementation SymbolicRegression.jl/PySR (Cranmer, 2020). Evolutionary algorithms represent different mathematical expressions as symbolic trees, and they generate alternative expressions by mutating previous expressions. For example, a possible mutation of the symbolic tree depicted in Figure 1 could be to replace the *sin* node with another operator, or replacing the terminal node *x1* with a new branch that grows from a new operator. Every expression is assigned a fitness value, and the expressions with higher fitness from the population are selected to reproduce so that the population can evolve over multiple generations.

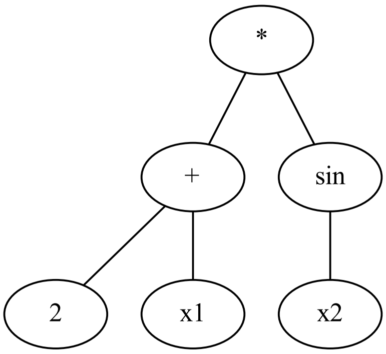

Figure 1

Symbolic tree that represents the equation $(2 + x1) \cdot \sin x2$.

A novel approach to perform symbolic regression is the use of Bayesian methods (Guimerà et al., 2020; Jin et al., 2019). These models use Markov chain Monte Carlo

[1] https://www.datarobot.com/nutonian/

(MCMC) samplers to explore the space of possible mathematical expressions and, similarly to the evolutionary algorithms, they represent the mathematical expressions as symbolic trees. In this case, instead of *mutating* the trees we say that MCMC samplers *jump* from one tree to another, although the modifications can be equivalent (e.g. replacing an operator or growing a new branch from where a terminal node used to be). However, the calculation of the probability of accepting a jump needs to satisfy a detailed balance that guarantees that the MCMC sampler will asymptotically converge to the posterior distribution of all possible mathematical expressions. The main advantage of Bayesian symbolic regression over evolutionary symbolic regression is that we can formally define all steps of the process using probability theory. For example, one of the main causes of concern with symbolic regression is that it can overfit a particular dataset if we do not penalize expressions that are excessively complex. In an evolutionary algorithm we can modify the fitness function to penalize complexity in some arbitrary way, but with a Bayesian algorithm we can specify different prior probabilities for different degrees of complexity.

The current state of the art in expression discovery are models that combine symbolic regression and deep neural networks to encode prior information about the structure of the target mathematical expressions. Cranmer et al. (2020) fit neural networks that encode prior scientific knowledge about the structure of the target system, and then they recover mathematical expressions with evolutionary algorithms from the networks. With a different approach, Udrescu and Tegmark (2020) use a neural network to discover hidden structure that is common in physical formulas (e.g. coherent units, symmetry). These methods require less data, generalize better to out-of-sample data and have better predictive performance than just the neural networks or evolutionary algorithms on their own. Bayesian symbolic regression is equivalent in scope with the evolutionary or the sparse regression algorithms, and we could use either of them in combination with neural networks or directly over a dataset.

In this report we build from Jin et al. (2019) Bayesian symbolic regression model. In the next section we describe the fundamentals of the original algorithm and we showcase the modifications that we have made to it in BayesianSR.jl[2], an easy-to-use and performant implementation in the Julia programming language (Bezanson et al., 2017). In the last

[2] https://github.com/cobac/BayesianSR

sections we compare the performance of the different modifications against the original algorithm and against an evolutionary algorithm.

Bayesian symbolic regression algorithm

Jin et al. (2019) Bayesian symbolic regression model is an additive model that consists of a linear combination of mathematical expressions represented as symbolic trees:

$$y_i = \beta_0 + \beta_1 \Psi_1(\boldsymbol{x}_i) + \cdots + \beta_K \Psi_K(\boldsymbol{x}_i) + \varepsilon_i \ ,$$

where Ψ_j is the j^{th} symbolic tree that represents a function of the features of the dataset $\boldsymbol{x}$, y is the outcome variable, β_j are the linear coefficients and ε are the residuals. The residuals follow a normal distribution $N(0, \sigma^2)$.

- $\boldsymbol{x}$ multiple predictors
- $\boldsymbol{y}$ one outcome variable
- Symbolic nodes are operators or features
    - No numbers, but linear operators
- Describe tree movements briefly
- Describe tree prior
- Describe briefly the process for each iteration
    - Tree proposal via jump
    - Parameter proposal from parameter prior
        - RJMCMC
    - OLS
    - Ratio calculation
- Explain the modifications that we are going to test
    - Mention fixes to the ratio calculation of the most recent version of their preprint

    - Centered proposals
    - Multiple chains
    - Symbolic simplification step
        - Reduce the size of posterior space
- Full model as an appendix
- Mention BayesianSR.jl package

Procedure

First of all we will compare the performance of the original algorithm against our implementation. Next, we will test the performance of the modifications we have made (i.e. symbolic simplification step, multi-chain sampling and both combined) against the basic algorithm and an evolutionary algorithm. The evolutionary algorithm we are going to use is SymbolicRegression.jl (Cranmer, 2020), a fast, parallel, distributed and open-source implementation.

To evaluate performance we need to measure three elements: accuracy of the models, complexity of the expressions and computational cost. Assessing accuracy is simple since we can use the Root Mean Squared Error (RMSE). Similarly, to assess complexity we can use the number of nodes of each expression. However, choosing how to measure computational cost is not straightforward because of two factors. The first one is that some algorithms are able to run in parallel while others are not. We have chosen to use real time instead of CPU time because we want to compare how efficient each algorithm is for a normal use case. When we run multiple chains on parallel we are interested in how fast the algorithm explores posterior space, and if we used CPU time we would be measuring parallel algorithms to perform multiple times worse than the non-parallel ones. The second factor is that to run our simulations we have to use a modified version of the Bayesian algorithms that periodically offload the MCMC chains from memory to avoid running out of memory. This step is unnecessary in a normal use case, but during the simulations it will produce significant overhead. To circumvent this issue so that the Bayesian simulations are comparable with the other algorithms we will use a standardized time unit. We will define a unit of time as the average time that a single iteration of the basic Bayesian algorithm takes during normal usage. We can calculate the average time that a single iteration takes for the rest of the algorithms and calculate proportionality their runtime in our custom units of time.

We are going to test all models with data generated from a standard set of functions (Expression 1) that have been used to benchmark other symbolic regression algorithms (Chen et al., 2015; Chen et al., 2016; Jin et al., 2019; Topchy, Punch, et al., 2001). We will use the same training and test conditions as originally reported by Jin et al. (2019): data generated without noise from $U(-3,3)$ for the training set, and from $U(-3,3)$, $U(-6,6)$ and $U(3,6)$ for three different test sets. We will run 50 simulations with datasets of 30 observations for 100,000 MCMC samples for each function. For the evolutionary model we will run a number of iterations that take a similar time than the Bayesian algorithms.

$$\begin{aligned}
f_1(x_0,x_1) &= 2.5x_0^4 - 1.3x_0^3 + 0.5x_1^2 - 1.7x_1\\
f_2(x_0,x_1) &= 8x_0^2 + 8x_1^3 - 15\\
f_3(x_0,x_1) &= 0.2x_0^3 + 0.5x_1^3 - 1.2x_1 - 0.5x_0\\
f_4(x_0,x_1) &= 1.5\exp(x_0) + 5\cos(x_1)\\
f_5(x_0,x_1) &= 6.0\sin(x_0)\cos(x_1)\\
f_6(x_0,x_1) &= 1.35x_0x_1 + 5.5\sin\left[(x_0-1)(x_1-1)\right] \qquad (1)
\end{aligned}$$

Lastly, we will use data from Wagenmakers et al. (2008) available on the R package rtdists (Singmann et al., 2020) to explore if we can recover mathematical expressions with a plausible scientific interpretation using both Bayesian symbolic regression and the evolutionary algorithm.

Results

- Measure of computational speed
  - Jin julia vs Jin python
- Show RMSE progression over time for all versions of the algorithms

- Acceptance rates
- Show complexity of the Bayesian expressions
    - Evolutionary report the Pareto frontier

Response time data showcase

- Test run of evolutionary vs Bayesian on the dataset
- Interpretability

Discussion

- Limitations of the comparisons
    - Unclear things
        - Effects of hyperparameters
        - Comparisons with sparse regression method
- Bayesian symbolic regression as an alternative to evolutionary algorithms
- Is it faster?
- Does it offer more control?
- Are the expressions more generalizable?
- Are the expressions more interpretable?
- The adoption of symbolic regression techniques in general in modeling / prediction use cases.

Materials

- Links to repository/osf

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Both, G.-J., Choudhury, S., Sens, P., & Kusters, R. (2021). Deepmod: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, *428*, 109985.

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, *113*(15), 3932–3937.

Chen, Q., Xue, B., Shang, L., & Zhang, M. (2016). Improving generalisation of genetic programming for symbolic regression with structural risk minimisation. *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 709–716.

Chen, Q., Xue, B., & Zhang, M. (2015). Generalisation and domain adaptation in gp with gradient descent for symbolic regression. *2015 IEEE congress on evolutionary computation (CEC)*, 1137–1144.

Cranmer, M. (2020). *Pysr: Fast & parallelized symbolic regression in python/julia*. Zenodo. https://doi.org/10.5281/zenodo.4041459

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., & Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. *CoRR*. http://arxiv.org/abs/2006.11287v2

Dandekar, R., Rackauckas, C., & Barbastathis, G. (2020). A machine learning-aided global diagnostic and comparative tool to assess effect of quarantine control in covid-19 spread. *Patterns*, *1*(9), 100145. https://doi.org/10.1016/j.patter.2020.100145

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A. G., Parizeau, M., & Gagné, C. (2012). Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, *13*(1), 2171–2175.

Guest, O., & Martin, A. E. (2020). How computational modeling can force theory building in psychological science. https://doi.org/10.31234/osf.io/rybh9

Guimerà, R., Reichardt, I., Aguilar-Mogas, A., Massucci, F. A., Miranda, M., Pallarès, J., & Sales-Pardo, M. (2020). A bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, *6*(5), eaav6971. https://doi.org/10.1126/sciadv.aav6971

Jin, Y., Fu, W., Kang, J., Guo, J., & Guo, J. (2019). Bayesian symbolic regression. *CoRR*. http://arxiv.org/abs/1910.08892v3

Lin, Q., Zhao, S., Gao, D., Lou, Y., Yang, S., Musa, S. S., Wang, M. H., Cai, Y., Wang, W., Yang, L., et al. (2020). A conceptual model for the coronavirus disease 2019 (covid-19) outbreak in wuhan, china with individual reaction and governmental action. *International journal of infectious diseases*, *93*, 211–216. https://doi.org/10.1016/j.ijid.2020.02.058

Robinaugh, D. J., Haslbeck, J. M. B., Ryan, O., Fried, E. I., & Waldorp, L. J. (2020). Invisible hands and fine calipers: A call to use formal theory as a toolkit for theory construction [PMID: 33593176]. *Perspectives on Psychological Science*, *0*(0), 1745691620974697. https://doi.org/10.1177/1745691620974697

Sahoo, S., Lampert, C., & Martius, G. (2018). Learning equations for extrapolation and control. *International Conference on Machine Learning*, 4442–4450.

Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, *324*(5923), 81–85.

Singmann, H., Brown, S., Gretton, M., & Heathcote, A. (2020). *Rtdists: Response time distributions* [R package version 0.11-2]. https://CRAN.R-project.org/package=rtdists

Topchy, A., Punch, W. F. et al. (2001). Faster genetic programming based on local gradient search of numeric leaf values. *Proceedings of the genetic and evolutionary computation conference (GECCO-2001)*, *155162*.

Udrescu, S.-M., & Tegmark, M. (2020). Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, *6*(16). https://doi.org/10.1126/sciadv.aay2631

van Rooij, I., & Baggio, G. (2020). Theory before the test: How to build high-verisimilitude explanatory theories in psychological science. *Perspectives on Psychological Science*, 1745691620970604. https://doi.org/10.1177/1745691620970604

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & McKoon, G. (2008). A diffusion model account of criterion shifts in the lexical decision task. *Journal of memory and language*, *58*(1), 140–159.

Appendix

Bayesian symbolic regression model specification

- Fixes from the 2020 version of their preprint

$$R = \frac{f\left[y|\mathrm{OLS}\left(\boldsymbol{x}, S^{*}, \Theta^{*}\right), \Sigma^{*}\right] f(S^{*}) q(S^{(t)}|S^{*}) f(\Theta^{*}|S^{*}) p(\Sigma^{*})}{f\left[y|\mathrm{OLS}\left(\boldsymbol{x}, S^{(t)}, \Theta^{(t)}\right), \Sigma^{(t)}\right] f(S^{(t)}) q(S^{*}|S^{(t)}) f(\Theta^{(t)}|S^{(t)}) p(\Sigma^{(t)})} \tag{2}$$

$$R = \frac{f\left[y|\mathrm{OLS}\left(\boldsymbol{x}, S^{*}, \Theta^{*}\right), \Sigma^{*}\right] f(S^{*}) q(S^{(t)}|S^{*}) f(\Theta^{*}|S^{*}) p(\Sigma^{*}) h\left(U^{*}|\Theta^{*}, S^{*}, S^{(t)}\right)}{f\left[y|\mathrm{OLS}\left(\boldsymbol{x}, S^{(t)}, \Theta^{(t)}\right), \Sigma^{(t)}\right] f(S^{(t)}) q(S^{*}|S^{(t)}) f(\Theta^{(t)}|S^{(t)}) p(\Sigma^{(t)}) h\left(U^{(t)}|\Theta^{(t)}, S^{(t)}, S^{*}\right)} \cdot \left|\frac{\partial j\left(\Theta^{(t)}, U^{(t)}|S^{(t)}, S^{*}\right)}{\partial\left(\Theta^{(t)}, U^{(t)}\right)}\right| \tag{3}$$