Accelerating Understanding of Scientific Experiments with End to End Symbolic Regression

**Nikos Aréchiga, Francine Chen, Yan-Ying Chen,
Yanxia Zhang, Rumen Iliev, Heishiro Toyoda, Kent Lyons**
Toyota Research Institute
Los Altos, CA
{nikos.arechiga, francine.chen, yan-ying.chen, yanxia.zhang, rumen.iliev, heishiro.toyoda, kent.lyons}@tri.global

Abstract

We consider the problem of learning free-form symbolic expressions from raw data, such as that produced by an experiment in any scientific domain. Accurate and interpretable models of scientific phenomena are the cornerstone of scientific research. Simple yet interpretable models, such as linear or logistic regression and decision trees often lack predictive accuracy. Alternatively, accurate blackbox models such as deep neural networks provide high predictive accuracy, but do not readily admit human understanding in a way that would enrich the scientific theory of the phenomenon. Many great breakthroughs in science revolve around the development of parsimonious equational models with high predictive accuracy, such as Newton's laws, universal gravitation, and Maxwell's equations. Previous work on automating the search of equational models from data combine domain-specific heuristics as well as computationally expensive techniques, such as genetic programming and Monte-Carlo search. We develop a deep neural network (MACSYMA) to address the symbolic regression problem as an end-to-end supervised learning problem. MACSYMA can generate symbolic expressions that describe a dataset. The computational complexity of the task is reduced to the feedforward computation of a neural network. We train our neural network on a synthetic dataset consisting of data tables of varying length and varying levels of noise, for which the neural network must learn to produce the correct symbolic expression token by token. Finally, we validate our technique by running on a public dataset from behavioral science.

Introduction

Accurate and interpretable models of real-world phenomena are the cornerstone of scientific progress. An accurate model makes high-quality predictions of the phenomenon under study. The recent rise of deep learning yields a general methodology for the production of high-accuracy prediction models: fit a deep neural network that accurately predicts the dependent variables of an experiment given values of the independent values. A key drawback of this approach, however, is that the deep neural network is not interpretable, in the sense that it is not feasible for a scientist to inspect the network and compose its model with other models from scientific theory.

Instead, we would like a model that can provide a parsimonious equational form that models the data under consideration. The general problem of inferring symbolic equations from data is known as *symbolic regression*. The general problem of symbolic regression is computationally difficult because equations are strings of symbols, and the number of possible strings grows exponentially with the length of the string. A direct attempt to enumerate and test all possible equational strings would take longer than the age of the universe to find the equation of interest (Udrescu and Tegmark 2020).

There is a rich of body of work around symbolic regression with metaheuristics such as sparse regression(Brunton, Proctor, and Kutz 2016; Quade et al. 2018) and genetic programming (Searson, Leahy, and Willis 2010; Pitzer and Kronberger 2021), including the commercial software tool Eureqa (Dubčáková 2011). Such metaheuristics, however, are computationally expensive.

In this work, we develop a neural network, called MACSYMA, to address the symbolic regression problem as an end-to-end supervised learning problem. We use supervised learning to map directly from experimental data tables to symbolic equations that describe that data. The computational complexity of our approach is reduced to the complexity of doing a feedforward pass of a neural network. Our training set accounts for variable length as well as noisy datasets. Our existing model has an output of maximum length, but in future work we will generalize our approach to use a recurrent neural network that is capable of producing sequences of any length. We demonstrate our approach on a public dataset from behavioral science that illustrates prospect theory (Kahneman and Tversky 1979).

Related Work

The problem of synthesizing symbolic expressions from raw data, commonly referred to as symbolic regression, has a rich history within the research community. Established approaches rely on the use of computationally expensive genetic algorithms. In this section, we review some recent work on symbolic regression.

The work in (Udrescu and Tegmark 2020) describes a physics-inspired method for symbolic regression, in which data is analyzed by a multistage process involving heuristics such as dimensional analysis, searching for symmetries, and others. In contrast, our work does not rely on manually encoded heuristics or domain-specific transformations, but instead directly transforms from raw data to a symbolic form.

In (Iten et al. 2020), the authors develop a neural network approach to modeling dynamic equations, predicting the evolution of dynamical systems over time. Their approach is similar to ours in the use of neural networks, but distinct in that we do not limit our attention to dynamical systems, but merely provide data of given variables and request that the model extract symbolic relationships.

(Cranmer et al. 2020) uses neural networks with a graph structure, intended to model the interactions between particles and forces, and uses a genetic programming package to extract the symbolic model. The usage of graph neural networks is intended to bias the model to produce symbolic expressions that are amenable to model sets of particles subject to forces, such as gravity or electromagnetism. In contrast, our approach completely elides the need for computationally expensive genetic programming techniques and is not limited to application domains with force-particle interactions.

In (Kusner, Paige, and Hernández-Lobato 2017), the authors train a variational autoencoder (VAE) over strings of equations. The VAE consists of an encoder network and a decoder network. The encoder network encodes equation strings into a latent space, which the decoder learns to reconstruct into equation strings. After training, the task of symbolic regression is performed by optimizing over the latent space to find a point that decodes to an equation that minimizes error over the given data. This optimization is computationally expensive.

An alternative approach in (Brence, Todorovski, and Džeroski 2021) models equations by a context-free grammar. Each production rule of the grammar is given a probability, and equations are produced through Monte Carlo search.

Both the latent-space optimization of (Kusner, Paige, and Hernández-Lobato 2017) and the Monte Carlo search of (Brence, Todorovski, and Džeroski 2021) are computationally expensive. In contrast, our approach performs inference by a feedforward pass of a neural network.

Methodology

The usage of our system is illustrated in Figure 1. First, a data table is collected from an experiment. This data table has columns corresponding to the independent variables as well as the dependent variable. For example, suppose an experiment contains the independent variables x_1 and x_2, and the goal is to obtain an equational model for the effect of these variables on the dependent variable y. Then, the data table consists of observations of x_1, x_2, and y. Next, this data table is fed into our neural network MACSYMA, which outputs an equation, for example $y = (w_1x_1 + w_2x_2)(w_3x_1 + w_4x_2)$, where $\{w_1, w_2, w_3, w_4\}$ are parameters of the equation. The final step is to perform a conventional regression procedure to fit values for the parameters w_i, resulting in an equational model of the data table.

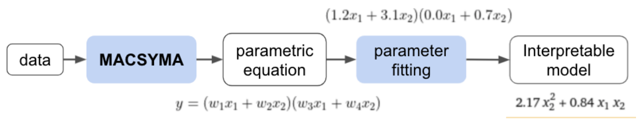
$(1.2x_1 + 3.1x_2)(0.0x_1 + 0.7x_2)$
data
MACSYMA
parametric equation
parameter fitting
Interpretable model
$y = (w_1x_1 + w_2x_2)(w_3x_1 + w_4x_2)$
$2.17\, x_2^2 + 0.84\, x_1\, x_2$

Figure 1: Usage flowchart. From a (possibly noisy) raw data table, MACSYMA produces a symbolic equation with unknown parameters w_i (in this case a quadratic form). A downstream regression task fits the parameters, and the equation can be simplified by off-the-shelf algebra software.

Synthetic dataset

To train the model, we developed a dataset that consists of noisy tabular data and the symbolic expression that corresponds to it. We generated this dataset by first collecting a series of symbolic expressions, including linear and polynomial equations, the min, max, argmin and argmax functions, and transcendental functions, such as exponentials, logarithms, sigmoids, sines, and cosines. In future work, we will add equations from the public dataset in (Udrescu and Tegmark 2020).

Each symbolic expression contains parameters and variables. The variables represent placeholders for the data to be modeled, and the parameters represent values to be fitted by the downstream regression task. Figure 2 illustrates the process for generating the dataset. We begin with a corpus of parametric equations. Each equation contains placeholders for numeric parameters as well as placeholders for data variables. Next, we randomly select values for the parameters of the equations. Then, we generate a data table for the equation by repeatedly plugging random values into the data variables and evaluating the equation. We allow for data tables of varying length, but not to exceed a total of 200 numerical values, including all dependent variable values as well as dependent variable values. Finally, we add Gaussian noise to the values of the dependent variables, to model the effect of measurement noise. For each equation, this process is repeated with ten different noise levels and with different randomly selected parameter instantiations to complete 5000 pairings between noisy data tables and their corresponding equations. By using different parameter instantiations, we encourage MACSYMA to learn the general mapping between data and its symbolic structure, generalizing across parameter values.

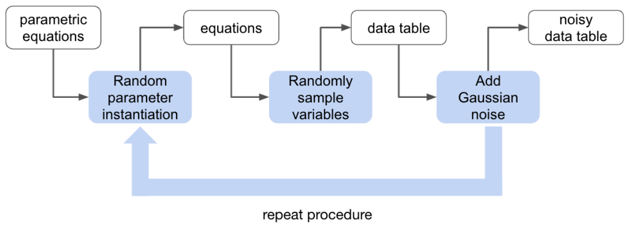

Figure 2: Data generation flowchart

Neural network architecture

Figure 3 diagrams the neural network architecture. The input vector consists of 201 elements. The first element is the number of data variables that appear in the equation, and the remainder of the vector is obtained by stacking the columns of the data table. Since the data tables have variable length, the stacked columns are repeated multiple times and then truncated to yield a vector of length 201. The network has 10 layers with 200 units each, and the output layer has $27,655$ elements, which is the number of elements needed to represent the longest equation in our corpus. The nonlinearity at all layers are ReLUs except for the last layer, which has a sigmoid nonlinearity. The outputs are interpreted as binary bits, which are decoded as one-hot encodings of the tokens in an equation. The loss function used is binary cross-entropy.

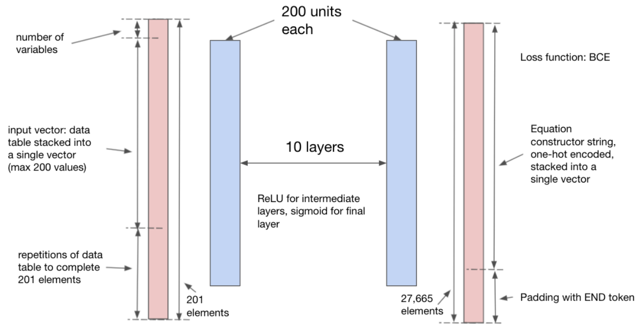
200 units each
number of variables
Loss function: BCE
input vector: data table stacked into a single vector (max 200 values)
10 layers
Equation constructor string, one-hot encoded, stacked into a single vector
ReLU for intermediate layers, sigmoid for final layer
repetitions of data table to complete 201 elements
201 elements
27,665 elements
Padding with END token

Figure 3: Neural network architecture

Model training

We trained for 100 epochs using a batch size of 64 training stimuli. Each training stimulus consists of an input data table and an output equation. Figure 4 shows the training loss, and Figure 5 shows the cross-validation loss. We continued training past the point where cross-validation error was increasing because the percentage of formulas that parsed correctly increased past the point at which cross-validation loss was minimal. In future work, we will investigate the trade-offs between these two metrics, and their effect on the performance of the equations generated.

An important challenge in the use of neural network models for symbolic regression is whether the model will efficiently learn the grammar of logical formulas. Previous work sidesteps this issue, for example in the use of a grammar-based checker which selects the most likely token that produces a grammatically correct formula, and backtracks when no such token exists (Kusner, Paige, and Hernández-Lobato 2017). Other approaches, such as (Brence, Todorovski, and Džeroski 2021), make the grammar of the logical formulas the basic scaffolding of the generative model, and only allow learning probabilities with which each production rule is applied. Approaches that rely on genetic programming to construct the final formula, such as (Dubčáková 2011) (and also (Cranmer et al. 2020), since the formula is extracted from the graph neural network by

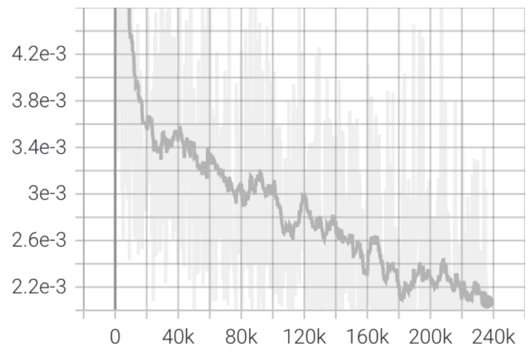

Figure 4: Training curve at each batch.

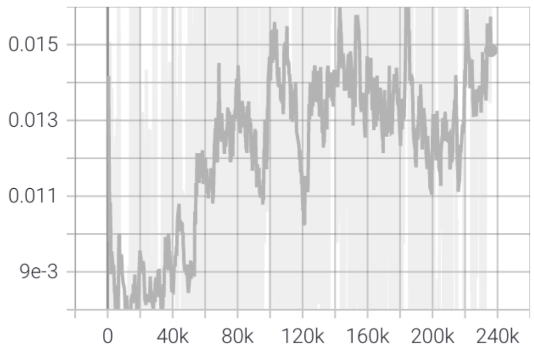

Figure 5: Validation curve. We continued training past the point where the validation loss was increasing because the percentage of parsable formulas continued to increase. We will investigate the trade-off between validation loss and percentage of parsable formulas in future work

a genetic program) can constrain the search space by only considering solutions that parse correctly.

Our approach, however, does not strictly enforce the grammar of logical formulas, and hence we rely on the ability of the neural network to learn the grammar. To evaluate the ability of the network to learn the grammar of logical formulas, Figure 6 demonstrates a plot of the network's ability to produce formulas that parse correctly. This is done by attempting to parse the formulas produced every 100 training steps in a batch of 64. The y-axis represents the percentage of formulas that parsed correctly. The percentage of formulas that parse correctly seems to plateau around 80%Ṫo the best of our knowledge, we are the first group to evaluate the ability of a neural network without grammatical scaffolding to produce equations that parse correctly. In the future, we will investigate the use of beam search, both to increase the percentage of formulas that parse correctly and to allow a human to interactively inspect several candidate formulas. The human interactivity could enable a scientist to select formulas with domain-specific meaning.

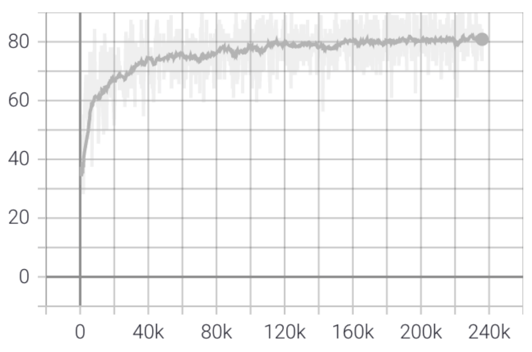

Figure 6: Percentage of formulas that parse correctly at each batch.

Results on an unseen dataset

As a case study, we use a subset of the Social Expected Utility (SEU) dataset (Iliev 2021). This dataset models a series of gambles presented to a survey participant. Each gamble corresponds of two possibilities, the first with a possibility of attaining a dollar value V_1 with probability p_1 versus a possibility of attaining dollar value V_2 with probability p_2.

The SEU dataset contains variations on this survey type, including ones in which the participant will benefit from one gamble, and the next survey participant will benefit from the second gamble. In this case study, we focus only on the case when both gambles are for the survey participant, and also both possible values are non-negative, $V_1 \geq 0$ and $V_2 \geq 0$.

One way to model this type of choice is to represent the *subjective value* (SV) of each alternative (p_i, V_i) by Equation 1 (Yu et al. 2017),

$$SV_i = p_i V_i^{\alpha} \tag{1}$$

where α varies from person to person, and is commonly interpreted as a *risk aversion parameter*. (Higher α is interpreted as a larger risk tolerance, (Yu et al. 2017)). A direct way to model the choice of alternatives is to simply select the choice with the larger subjective value,

$$\text{choice} = \arg\max(SV_1, SV_2) \tag{2}$$

An alternative (Yu et al. 2017) is to add an additional "noise" parameter, β, which scales a sigmoid. Equation 3 yields a number between 1 and 2, and may be rounded to give a prediction of the choice the participant will make.

$$\text{choice} = \sigma(\beta(SV_1 - SV_2)) + 1 \tag{3}$$

The structure of both Equation 2 and Equation 3 appear in the training corpus of MACSYMA. The SEU dataset is used for testing only. When we run MACSYMA on the SEU dataset, we recover Equation 2, which is one of the well-known theory-based models.

Discussion and future work

An important drawback of our existing neural network architecture is that there is a maximum size to the formula that can be produced. Furthermore, using a static, non-recurrent architecture means that increasing the maximum formula length means that the size of the neural network will increase exponentially, because all of the layers are fully connected. In future work, we will explore the use of a recurrent architecture to address these issues. Additionally, we will benchmark our approach against the public AI Feynman dataset (Udrescu and Tegmark 2020), extending the range of equations that are supported and enabling comparison against other approaches. We will also implement beam search at the output of the neural network to enable a human operator to interactively search for formulas that may have a specific interesting structure to the domain.

References

Brence, J.; Todorovski, L.; and Džeroski, S. 2021. Probabilistic Grammars for Equation Discovery. *Knowledge-Based Systems*, 224: 107077.

Brunton, S. L.; Proctor, J. L.; and 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.

Cranmer, M.; Sanchez-Gonzalez, A.; Battaglia, P.; Xu, R.; Cranmer, K.; Spergel, D.; and Ho, S. 2020. Discovering Symbolic Models from Deep Learning with Inductive Biases. *arXiv:2006.11287 [astro-ph, physics:physics, stat]*.

Dubčáková, R. 2011. Eureqa: Software Review. *Genetic Programming and Evolvable Machines*, 12(2): 173–178.

Iliev, R. 2021. Social Expected Utility Dataset. https://github.com/TRI-MAC/SEU.

Iten, R.; Metger, T.; Wilming, H.; del Rio, L.; and Renner, R. 2020. Discovering Physical Concepts with Neural Networks. *Physical Review Letters*, 124(1): 010508.

Kahneman, D.; and Tversky, A. 1979. Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47(2): 263.

Kusner, M. J.; Paige, B.; and Hernández-Lobato, J. M. 2017. Grammar Variational Autoencoder. *arXiv:1703.01925 [stat]*.

Pitzer, E.; and Kronberger, G. 2021. Smooth Symbolic Regression: Transformation of Symbolic Regression into a Real-Valued Optimization Problem. *arXiv:2108.03274 [cs]*, 9520: 375–383.

Quade, M.; Abel, M.; Kutz, J. N.; and Brunton, S. L. 2018. Sparse Identification of Nonlinear Dynamics for Rapid Model Recovery. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6): 063116.

Searson, D. P.; Leahy, D. E.; and Willis, M. J. 2010. GPTIPS:An Open Source Genetic Programming Toolbox For Multigene Symbolic Regression. *Hong Kong*, 4.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A Physics-Inspired Method for Symbolic Regression. *Science Advances*, 6(16): eaay2631.

Yu, L. Q.; Lee, S.; Katchmar, N.; Satterthwaite, T. D.; Kable, J. W.; and Wolf, D. H. 2017. Steeper Discounting of Delayed Rewards in Schizophrenia but Not First-Degree Relatives. *Psychiatry Research*, 252: 303–309.